22nd OpenMath Workshop

Editor: James H Davenport
University of Bath

July 9th 2009
Grand Bend Ontario
Draft of book to be published by
the University of Bath Press and the OpenMath Society
ISBN of final book: 978-1-86197-172-2

Table of Contents

A. Contributed Papers

B. Content Dictionary Descriptions

OpenMath in SCIEnce: Evolving of Symbolic Computation Interaction

Sebastian Freundt[1], Peter Horn[2], Alexander Konovalov[3], Sylla Lesseni[1], Steve Linton[3], and Dan Roozemond[4]

[1] Fakultät II - Institut für Mathematik, Technische Universität Berlin, Berlin, Germany, {freundt|lesseni}@math.tu-berlin.de
[2] Fachbereich Mathematik, Universität Kassel, Kassel, Germany, horn@math.uni-kassel.de
[3] School of Computer Science, University of St Andrews, Scotland, {alexk|sal}@mcs.st-and.ac.uk
[4] Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, Netherlands, d.a.roozemond@tue.nl

Abstract. We present SCSCP – the Symbolic Computation Software Composability Protocol. SCSCP is a remote procedure call framework for computational algebra systems in which both data and protocol instructions are encoded in the OpenMath language. We present SCSCP implementations in several CASes and other SCSCP-compliant applications and APIs, developed with the support of the EU FP6 project "SCIEnce – Symbolic Computation Infrastructure for Europe".

1 Combining Symbolic Computation Systems

Many research problems which could be tackled with computer algebra systems (CASes) cannot be solved within a single system or could be solved much faster if a combination of two or more CASes would allow performing each step in the fastest available implementation.

Examples include number theory computations in a specialized system, symbolic calculations on generic character tables, computations on large finite state automata, Gröbner computations that are faster in one system than in another, libraries or plug-ins that are available in a system on Linux but not on Windows, etc.

In this paper we are giving an overview of the SCIEnce project activity to develop a framework which enables efficient and reliable combining of CASes for such purposes. Another area the SCIEnce project hopes to bring improvement to is that of libraries and databases of mathematical objects that may be stored in some universal format accessible to many systems.

Since the OpenMath standard emerged, a lot of work was done on combining CASes, e.g. the MONET project [8] and the various translators (phrasebooks) produced by RIACA [14] (see also the OpenMath webpage [10] for a detailed account of OpenMath software and tools). However, the approach has often been

to have a system provide OpenMath by creating wrapper software that communicates with the system in the background and that performs the translation from OpenMath to the internal syntax and vice versa.

The approach taken in the SCIEnce project, however, is to build the OpenMath support into the systems themselves instead of creating a wrapper, which yields a much more robust implementation. Also, unlike in another well-known project in this area, namely Sage [17], the approach in SCIEnce is not to subordinate the packages from an integrating system, but to define an interface that any system can implement to provide a way to use the capabilities of other systems within a familiar environment.

We therefore set up a project aiming to define standards and construct an extensible framework which would:

- ensure seamless communication between CASes, both local and remote;
- use a universal format not relying on the particular input/output format for each system;
- ensure that each system implementing the standard can immediately offer services to and consume services from other such systems.

2 SCSCP and its CDs

To simplify the communication between the various CASes, we have developed a protocol called the "Symbolic Computation Software Composability Protocol", abbreviated SCSCP [3, 2]. This protocol does not only enable the computation of simple commands in a different system or on a different machine, but it will also serve as a means of conveying constituents of larger, more complex, computations.

The key features of SCSCP are:

- Mathematical data are encoded in OpenMath;
- The protocol messages are encoded in OpenMath as well, so that participating systems need to support only one language;
- The OpenMath support is wired directly into the joining systems. This is much more robust, easier to create, and faster than the usual practice of producing wrapping programs to enable OpenMath support.

In particular, the protocol messages are in the OpenMath language, and its TCP-sockets based implementation uses XML processing instructions to delimit these messages and convey small pieces of information on a higher level. Communication takes place using port 26133, reserved for SCSCP by the Internet Assigned Numbers Authority (IANA).

SCSCP does not require statefulness on behalf of the server, although it does offer support for working with so-called "remote objects." Such object can be created on the server (thus, changing its state) and used in further computations as arguments of procedure calls.

At the moment of writing the protocol has reached version 1.3 and both client and server implementations exist in GAP, KANT, Maple, and MuPAD.

We will detail these implementations in Section 3, except for the Maple system which plans to announce its tools elsewhere at a later stage.

The protocol is also supported by TRIP, a general computer algebra system dedicated to celestial mechanics, using an own publicly available implementation of SCSCP [4]. Moreover, we have developed a Java library `org.symcomp.scscp` to facilitate third party developers in exposing their own applications using SCSCP. We provide details on this library in Section 4. Additionally, while some of our SCSCP implementations provide straightforward functionality for parallel computations, another result of the SCIEnce project is the SymGrid-Par middleware, which orchestrates computational algebra components into a parallel application that uses SCSCP for internal communication [19].

Apart from two OpenMath Content Dictionaries accompanying the SCSCP protocol [15, 16], several other Content Dictionaries were developed in the project, concerning, for example, polynomial factorization and efficient OpenMath representations of matrices, number fields and orders in number fields. We have submitted these content dictionaries separately to the OpenMath 2009 workshop.

As a simple example, we demonstrate a simple SCSCP session on the server. The server is running GAP and provides a procedure to identify a finite group in the GAP Small Groups Library. After the server receives an incoming connection, it replies with the connection initiation message. After that, the client replies with its preferred version, and the server confirms this version to the client.

```
S: <?scscp service_name="GAP" service_version="4.dev" service_id="
     localhost:26133:7617" scscp_versions="1.0 1.1 1.2 1.3" ?>
C: <?scscp version="1.3" ?>
S: <?scscp version="1.3" ?>
```

Then the client sends the procedure call to identify the cyclic group of order two given as permutation group:

```
C: <?scscp start ?>
   <OMOBJ>
       <OMATTR>
           <OMATP>
               <OMS cd="scscp1" name="call_id"/>
               <OMSTR>scscp.symcomp.org:26133:7617:eBFyqFae</OMSTR>
               <OMS cd="scscp1" name="option_return_object"/>
               <OMSTR></OMSTR>
           </OMATP>
           <OMA><OMS cd="scscp1" name="procedure_call"/>
               <OMA><OMS cd="scscp_transient_1" name="WS_IdGroup"/>
                   <OMA><OMS cd="permgp1" name="group"/>
                       <OMS cd="permutation1" name="right_compose"/>
                       <OMA><OMS cd="permut1" name="permutation"/>
                           <OMI>2</OMI>
                           <OMI>1</OMI>
                       </OMA>
                   </OMA>
               </OMA>
           </OMA>
       </OMATTR>
   </OMOBJ>
   <?scscp end ?>
```

The server responds that the group has catalogue number $[2, 1]$:

```
S: <?scscp start ?>
   <OMOBJ>
       <OMATTR>
           <OMATP>
               <OMS cd="scscp1" name="call_id"/>
               <OMSTR>scscp.symcomp.org:26133:7617:eBFyqFae</OMSTR>
           </OMATP>
           <OMA><OMS cd="scscp1" name="procedure_completed"/>
               <OMA><OMS cd="list1" name="list"/>
                   <OMI>2</OMI>
                   <OMI>1</OMI>
               </OMA>
           </OMA>
       </OMATTR>
   </OMOBJ>
   <?scscp end ?>
```

After that the client closes the connection, and the server is ready to accept new procedure calls.

3 Computer Algebra Systems

In this section we give a brief overview of the status of the SCSCP implementation in various systems.

3.1 GAP

In the GAP system, the support of OpenMath and SCSCP is implemented in two GAP packages with the same names.

The OpenMath package [1] provides an OpenMath phrasebook for GAP: it is responsible for the conversion from OpenMath to GAP and vice versa and reading/writing OpenMath objects from/to streams. The package provides a framework, allowing users to extend it with private content dictionaries.

The SCSCP package [7] implements the Symbolic Computation Software Composability Protocol on top of the GAP packages OpenMath, IO and GAP-Doc. The package has two main components: server and client. The server may be started interactively from the GAP session or as a GAP daemon. When the server accepts a connection from the client, it starts the "accept-evaluate-return" loop:

- accepts the `"procedure_call"` message;
- performs lookup of the appropriate GAP function;
- evaluates the result (or produces a side-effect);
- returns the result in the `"procedure_completed"` message or returns an error in the `"procedure_terminated"` message.

The SCSCP client performs the following basic actions:

- establishes connection with the specified server at the specified port;
- sends the `"procedure_call"` message to the server;

- waits for the result of the computation or returns to pick it up later;
- fetches the response, extracting the result from the `"procedure_completed"` message or entering the break loop in the case of the `"procedure_terminated"` message.

On top of this functionality we built a set of instructions for parallel computations using the SCSCP framework, allowing to send several procedure calls in parallel and then collect all results or pick up the first available result, and implemented the master-worker parallel skeleton.

To give the users an opportunity to test the package we are running a demo SCSCP server accessible at `chrystal.mcs.st-andrews.ac.uk`, port 26133. It is working under the development version of the GAP system and a selection of currently redistributed GAP packages. See the package homepage [7] for further information, downloads and documentation with examples.

3.2 KANT

The KANT system provides two main packages for SCSCP support:

- `libkant` package which contains the core functionality of the KANT system;
- `autokash` which consists of the KANT SCSCP server, a simple client and a server. The server is started by running the KANT/KASH daemon, named `kashd`, which is the main binary in the `autokash` package.

Here are the different steps when the connection from a client is accepted by the server:

- a socket is open and the SCSCP message comes in;
- the `"procedure_call"` part of the message is extracted;
- a table of xpaths is matched against the message and a callback function is looked up;
- after evaluation, the result (or an error message in case of error) is sent back to the client.

We are running two KANT SCSCP servers accessible at port 26133 at addresses `issel.math.tu-berlin.de` and `stirling.math.tu-berlin.de` and running under the recent development version of the KANT system. The users can also download the `autokash` package which contains `libkant` library from [6].

The OpenMath support is implemented in the `autokash` package. It is contained in the `openmath.la` library. To use that library, one should load it when running the KANT/KASH daemon `kashd`. See the public homepage [6] for more information.

The KANT SCSCP Client Shell: kapy We are developing at the present moment a KANT SCSCP client shell, named kapy. It is written in python and fully supports the SCSCP protocol. Also, the idea of using kapy is to ease manual typing when handling OpenMath objects. To connect from kapy to a running KANT SCSCP server, we need at least python version 2.5 and the script

`kapy.py`. The call `cas = kapy.connect(`*host*`,` *port*`)` will establish the connection with the SCSCP server running at the appropriate host and port. From then, one can create the openmath objects like OMI, OMF, OMSTR, OMV using:

- `cas.compute(kapy.omi(`*int*`))` for the intergers;
- `cas.compute(kapy.omf(`*float*`))` for the floats;
- `cas.compute(kapy.omf(`*str*`))` for the strings;
- `cas.compute(kapy.omv(`*var*`))` for the unused variables since the KANT system can not handle the symbolic objects.

The openmath objects OMS and OMA are constructed as follows:

- `cas.compute(kapy.oms(`*cdname*`,` *symbolname*`))` for OMS;
- `cas.compute(kapy.oma(`*cdname*`,` *symbolname*`,` *args*`))` for OMA.

We can also store the result in a variable to reuse it later or in other computations. Finally, since kapy is written in python, it is natural to be able to convert basic OM objects like OMI, OMF, OMSTR to the python representation using a function named `PyConvert`.

3.3 MuPAD

There are two main aspects for MuPAD SCSCP support:

- `OpenMath` MuPAD package;
- SCSCP server wrapper for MuPAD.

While the former offers the ability to parse, generate, and handle OpenMath in MuPAD, and to consume SCSCP services, the latter enables access to MuPADs mathematical abilities as an SCSCP service. Sadly, however, the current MuPAD end-user license agreement does not generally allow this. Therefore, below we concentrate on the `OpenMath` package.

To use the package, download it from [9] and put it into your `PACKAGEPATH`. It can then be loaded using `package("OpenMath")`. Afterwards, documentation is available through `OpenMath::doc()`.

The OpenMath Elements To represent the different OpenMath tags, there are the following constructors:

- `OpenMath::Apply(`*head*`, [`*params*`])` – expands to a function call of *head* on *params*;
- `OpenMath::Bind(`*head*`, [`*vars*`],` *expr*`)` – expands to a function call of *head* on *vars* and *expr*;
- `OpenMath::Error(`*head*`, [`*params*`])` – expands to an error textually containig *head* and *params*;
- `OpenMath::Float(`x`)` – expands to a `DOM_FLOAT`;

- `OpenMath::Integer(`*i*`)` – expands to a `DOM_INT`;
- `OpenMath::Object(`*o*`)` – expands to *o*;
- `OpenMath::Reference(`*id*`)` – expands to either the element with the given *id* or an `error`;
- `OpenMath::String(`*str*`)` – expands to the given `DOM_STR`;
- `OpenMath::Symbol(`*cdname*`, `*name*`)` – expands to either some MuPAD object or the MuPAD identifier ‘*cdname*`.`*name*‘;
- `OpenMath::Variable(`*name*`)` – expands to an unused `DOM_IDENT`.

All constructors accept an optional last argument *id* to set the id. A tree of these objects may be translated to MuPAD objects by calling `expand` on it. It may be necessary to call `eval` to get an actual result.

Calling `OpenMath::toXml` on an OpenMath tree gives a tree of `adt::XML` nodes representing it. This can then printed or converted to an XML string. All these constructors have a `doc` slot, so you can get further information by calling, e. g., `OpenMath::Apply::doc()`.

The OpenMath Parser In this domain, there are two functions available to turn an OpenMath XML string into a tree of `OpenMath::` objects as above:

- `OpenMath::parse(`*str*`)` – parses the string *str*;
- `OpenMath::parseFile(`*fname*`)` – reads and parses the file named *fname*.

Generating OpenMath With `generate::OpenMath`, a MuPAD expression can be converted into its OpenMath representation. Internally, the above mentioned `OpenMath` elements are used to assemble the result.

The result of the call to `generate::OpenMath` is always wrapped in an `OpenMath::Object`. To obtain the OpenMath representation without the wrapping `OMObject`, one can simply use `OpenMath(...)`.

SCSCP Client Connection By the call `s := SCSCP(`*host*`, `*port*`)` an SCSCP connection object is created, which can then be used to send commands to the SCSCP server. Note that the actual connection is initiated on construction by starting the java program WUPSI (see 4.4) which is bundled into the `OpenMath` package. It is using an asynchronous file system based message exchange mode and thus can be used to do computations in the background.

To actually let the server compute something, one uses `s::compute(...)` or, equivalently `s(...)`. Note that it may be necessary to wrap the parameter in `hold(...)` to prevent premature evaluation in MuPAD.

To use the connection asynchronously, the commands `send` and `retrieve` are used: `a := s::send(...)` returns an integer which may be used to identify the computation and to retrieve the result later with `s::retrieve(a)`. `retrieve` returns `FAIL` if the result of the computation is not yet computed unless you specify a second parameter `TRUE`. In that case the call will block until the result is ready.

To disconnect the client after use (and before e. g. `reset`) one can use the command `s::close()` to stop the corresponding Java program, and thus clean up the associated resources. Obviously, when MuPAD exits, this is done automatically.

4 Java SCSCP API

This Java library is intended to enable third party developers to use SCSCP servers (i.e. have their own application acting as an SCSCP client), or easily expose their own applications as SCSCP servers (i.e. have their own application acting as an SCSCP server).

The library has two essential parts: The OpenMath library `org.symcomp.openmath` and the SCSCP implementation `org.symcomp.scscp`. We will detail these libraries in Sections 4.2 and 4.3, respectively. Furthermore, the library comes with several examples that should serve as a good starting point for the user.

We use these libraries ourselves as well: In the MuPAD client and server application (Section 3.3), in an experimental Magma server, in the Webproxy (Section 4.5) and in WUPSI (Section 4.4).

4.1 The Popcorn representation

When handling OpenMath objects, one frequently finds oneself typing and reading lots of `OMA`s, `OMS`s, and so on. This may lead one to the conclusion that humans were not designed to parse XML. That is why we decided to create an OpenMath representation taking this into account, and created POPCORN. It is an acronym standing for "Possibly Only Practical Convenient OpenMath Replacement Notation". For the sake of typographic beauty, we write it as "Popcorn".

We emphasize that Popcorn is merely an OpenMath representation that we consider convenient for humans, similar to the binary representation that is obviously more convenient for machines. Furthermore, if a two-dimensional environment such as a web browser is available, more sophisticated editors such as the MathDox formula editor are even better. However, we still think Popcorn is a valuable addition, e.g. for quick tests, command line applications, etc.

Popcorn is described in more details in the MKM 2009 paper [5].

4.2 `org.symcomp.openmath`

Although there are some Java OpenMath Libraries available [12, 13], these are older (last update in 2000 and 2004, respectively) and we disagreed with some of the design choices made.

We therefore created a new library that takes advantage of the recent developments in Java, such as annotations and generics, and we designed it from the ground up to be as easily extensible as possible. It provides many convenience classes and handy methods to traverse, construct, and analyze OpenMath trees.

Furthermore, it has completely transparent support for OpenMath Attributions, eliminating the need to handle these objects in any special way.

Import and export to OpenMath 2 XML, OpenMath 2 Binary, and Popcorn, and export to LaTeX are included. Moreover, to feed OpenMath data into other applications, it is often necessary to produce a specific format. This is wired into `org.symcomp.openmath` as *custom renderers*. We designed this part of the library in such a way that producing e. g. a renderer for the MAGMA language took only a few lines of code. Moreover, the LaTeX- and Popcorn-renderer are made using the same mechanism. These also give the user a great starting point for developing his/her own custom renderer.

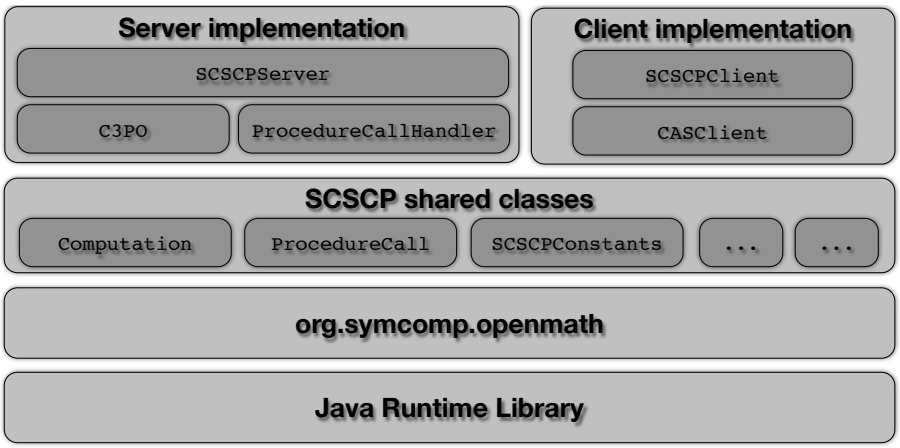

Fig. 1. The structure of the `org.symcomp` Java libraries

4.3 `org.symcomp.scscp`

As mentioned above, this library was designed to enable a third party developer to easily expose his or her own application using SCSCP. This library obviously uses `org.symcomp.openmath` for handling the OpenMath objects that are inherent to the SCSCP protocol.

The structure of the libraries is shown in Figure 1. Depending on the level of control a developer wants to have on the inner workings of the SCSCP protocol, he would extend a particular class.

For an application to use an SCSCP service such as GAP or MuPAD, a developer would typically subclass the `SCSCPClient`. She or he would then need to do little else than specify the host and port to connect to, phrase the relevant questions in OpenMath, and call the `compute` method.

If on the other hand someone wants to make his or her own application available for other SCSCP clients, she or he would typically subclass called the

`ProcedureCallHandler`. In its `handle` method, little else is required than interpreting the incoming OpenMath message, converting it to the internal format of the parent application, computing the solution, and converting the solution back to OpenMath.

4.4 WUPSI : a proof-of-concept example

The API described in this section allowed us to easily create WUPSI ("Wonderful Universal Popcorn SCSCP Interface"). It is a small command-line application that allows to connect to one or more SCSCP servers and issue computation requests.

It was designed with two main purposes in mind. Firstly, it serves as a great debugging tool for SCSCP implementations, as one simply enters OpenMath, and receives OpenMath back (possibly in the form of Popcorn). Secondly, it is an extensive example of how the Java libraries may be used, and could serve as a nice reference for those who want to use these libraries.

Apart from the example uses shown in Listing 2 many other more advanced functions are available, such as a poor-man's parallelization tool and the possibility to have WUPSI act as an SCSCP proxy server for connected systems.

4.5 WebProxy

The WebProxy is a Java application meant as an administration and orchestration console for one or more SCSCP compliant services. Whilst the WebProxy is a web application intended for direct user interaction, it also provides access to the capabilities of the connected CASes through SOAP and GET/POST interfaces.

5 Licensing and Availability

The GAP packages are distributed under the GNU Public License as well as the GAP system itself and are available from their sites and from the GAP homepage `http://www.gap-system.org`.

The KANT system and the KANT SCSCP packages are distributed under the GNU Public Licence and are available from the KANT website `http://www.math.tu-berlin.de/~kant/`.

The MuPAD package is not provided by the makers of MuPAD, SciFace Software, but by the University of Kassel. It is published [9] under an Apache 2 License and should be compatible with MuPAD 4 and above.

The SCSCP library `org.symcomp.scscp` and the `org.symcomp.openmath` library are released under the Apache 2 License. In February 2009 the first public release was made [11]. The libraries are available as binaries, source packages or they may be used as Maven dependencies. Available on the website is also a comprehensive (and continuously improving) API documentation. WUPSI will be available for download from [11] shortly.

```
WUPSI 1.2 -- Wonderful Universal Popcorn SCSCP Interface
(c) 2009 D. Roozemond & P. Horn

WUPSI[n/a]0> connect some.server:26139 as gap
# connected to 'some.server' on port '26139' using symbolic name 'gap'
# Service Info: service Name 'GAP', service version '4.dev'

WUPSI[gap]0> 126+2323*232
539062

WUPSI[gap]1> local $a := $_out0
# Stored this in local variable '$a':
539062

WUPSI[gap]2> connect 127.0.0.1:26134 as mupad
# connected to '127.0.0.1' on port '26134' using symbolic name 'mupad'
# Service Info: service Name 'MuPAD', service version '0.6.0-mupad
    -5.2.0'

WUPSI[mupad]2> output format latex
# switched output format to LATEX.

WUPSI[mupad]2> sum(1 .. infinity, lambda[$x -> 1/$x^2])
{\pi}^{2} \cdot \frac{1}{6}

WUPSI[mupad]3> output format popcorn
# switched output format to POPCORN.

WUPSI[mupad]3> local $p := 2^127-1
# Stored this in local variable '$p':
170141183460469231731687303715884105727

WUPSI[magma]4> use gap
# switched to system with symbolic name 'gap', service Name 'GAP',
    service version '4.dev'.

WUPSI[gap]4> $p-2^101*$a
168774498924748772136428072069291311103

WUPSI[gap]4> describe arith1.plus
# -- Description for 'arith1.plus' --
The symbol representing an n-ary commutative function plus.
# -- END description for 'arith1.plus' --
```

Listing 2. Using WUPSI

For the links to the most recent available downloads see the homepage of the SCIEnce project `http://www.symbolic-computation.org/`

6 Conclusions and future work

In this paper we presented SCSCP - a simple light-weight OpenMath-based remote procedure call framework, and gave an overview of SCSCP-compliant applications, represented by computer algebra systems, middleware and APIs. Further information and concrete examples may be found in (mostly available online) documentation for appropriate tools.

Among our future directions are developments of new content dictionaries additionally to those submitted to the OpenMath 2009 workshop, and increasing

the support of binary OpenMath format which seems inevitably needed in really large-scale computations.

We hope that the appearance of SCSCP and examples of its use in our CASes stimulate developers of other systems to support the OpenMath format to exchange mathematical data, and we hope more SCSCP-compliant software will become available in the future. We would like to strengthen this invitation by offering our support and advice to all interested parties.

References

1. M. Costantini, A. Konovalov and A. Solomon. *GAP package OpenMath*. Version 10.0, 2009. `http://www.cs.st-andrews.ac.uk/~alexk/openmath.htm`
2. S. Freundt, P. Horn, A. Konovalov, S. Linton and D. Roozemond. Symbolic Computation Software Composability. In *Intelligent Computer Mathematics, AISC/Calculemus/MKM 2008 proceedings*, Lecture Notes in Computer Science 5144/2008, Springer, p.285-295.
3. S. Freundt, P. Horn, A. Konovalov, S. Linton, D. Roozemond, Symbolic Computation Software Composability Protocol (SCSCP) specification, Version 1.3, 2009. `http://www.symbolic-computation.org/scscp/`
4. M. Gastineau, *SCSCP C Library - A C/C++ library for Symbolic Computation Software Composibility Protocol*, IMCCE, 2009, `http://www.imcce.fr/Equipes/ASD/trip/scscp/`
5. Peter Horn and Dan Roozemond. OpenMath in SCIEnce: SCSCP and POPCORN. To appear in *Intelligent Computer Mathematics, MKM 2009 proceedings.*
6. *KANT SCSCP Package.* `http://www.math.tu-berlin.de/~kant/kantscscp.html`
7. A. Konovalov and S. Linton. *GAP package SCSCP*. Version 1.1, 2009. `http://www.cs.st-andrews.ac.uk/~alexk/scscp.htm`.
8. *The MONET project.* `http://monet.nag.co.uk/monet/`
9. *MuPAD OpenMath Package.* `http://mupad.symcomp.org/`
10. *OpenMath.* `http://www.openmath.org/`
11. The `org.symcomp.openmath` and `org.symcomp.scscp` libraries: `http://java.symcomp.org/`
12. *PolyMath/OpenMath.* `http://pdg.cecm.sfu.ca/openmath/`
13. *RIACA OpenMath Library.* `http://www.mathdox.org/new-web/openmath.html`
14. *RIACA OpenMath Phrasebooks.* `http://www.mathdox.org/new-web/products.html`
15. Roozemond, D.: *OpenMath Content Dictionary: scscp1.* `http://www.win.tue.nl/SCIEnce/cds/scscp1.html`
16. Roozemond, D.: *OpenMath Content Dictionary: scscp2.* `http://www.win.tue.nl/SCIEnce/cds/scscp2.html`
17. *Sage.* `http://sagemath.org/`
18. *Symbolic Computation Infrastructure for Europe.* `http://www.symbolic-computation.org/`
19. A.D. Al Zain, P.W. Trinder, K. Hammond, A. Konovalov, S. Linton and J. Berthold. Parallelism without Pain: Orchestrating Computational Algebra Components into a High-Performance Parallel System. In *International Symposium on Parallel and Distributed Processing with Applications*, 2008, p.99–112.

The Intergeo File Format in Progress

Miguel Abánades[1], Francisco Botana[2], Jesús Escribano[3],
Maxim Hendriks[4], Ulrich Kortenkamp[5], Yves Kreis[6],
Paul Libbrecht[7], Daniel Marques[8], Christian Mercat[9]

[1] CES Felipe II - UCM, Aranjuez, Spain
[2] Universidad de Vigo, Spain
[3] Universidad Complutense de Madrid, Spain
[4] Eindhoven University of Technology, The Netherlands
[5] University of Education Karlsruhe, Germany
[6] University of Luxembourg, Luxembourg
[7] DFKI GmbH, Saarbrücken, Germany
[8] Maths for More (WIRIS), Barcelona, Spain
[9] I3M, Université Montpellier 2, France

Abstract. In this paper we describe the ongoing effort to specify a common file format for Interactive or Dynamic Geometry Systems (DGS). Our approach is based on the OpenMath standard, and uses its flexible extension mechanisms like Content Dictionaries.
We discuss the various design decisions, the Content Dictionaries that have been defined, as well as open questions to be resolved.

1 Introduction: The Intergeo Project

Interactive geometry is one of the most well known family of computer-based tools to support teaching of mathematics by means of personal explorations.

Intergeo (`http://inter2geo.eu`) is an eContent*plus* European project dedicated to the sharing of interactive geometry constructions across boundaries. It enables teachers and pupils all over Europe to share resources and experiences as tools for teaching, learning, and research.

Educational contents that were hard to access shall be made available, tagged with relevant topics and competency based metadata and categorised according to curricula, they are searchable and easily (re-)usable by everyone. It is our goal to offer them in a common interoperable format that this article describes.

For more information about the project, we refer to its website and the documentation available there, as well as [1,2].

A wide variety of DGSs exists. Before the Intergeo project, each system used incompatible proprietary file formats to store its data. Thus, most of the DGS makers have joined to provide a common file format that will be adopted either in the core of the systems or just as a way to interchange content.

The Intergeo file format is a file format designed to describe any construction created with a Dynamic Geometry System (DGS). Dynamic Geometry Systems, also called Interactive Geometry Systems, are programs that are used to experiment with geometric objects. A construction, a drawing consisting of geometric

elements, is displayed to the user. But it is not just a still picture. The construction is interactive, it reacts to user's input, who can move some of the elements with the mouse pointer. The whole construction is then recomputed according to the defining geometric relationships. For example, the circumcircle of a triangle could follow the three vertices of the triangle wherever they are dragged.

The Intergeo file format is based on three design decisions: the packaging as an archive with particular files inside it, the separation of the elements part describing the (static) initial geometry from the constraints part where the geometric relationships are expressed using OpenMath, and the use of OpenMath Content Dictionaries (CDs) to describe the elements and the constraints. These CDs are provisionally called the i2geo CDs.

This paper describes the ongoing specification effort of the aforementioned Content Dictionaries and the progress on implementation.

2 Review of pre-existing CDs

The primary and ambitious goal of OpenMath (`http://www.openmath.org`) is to develop a standard for representing mathematical objects with their semantics. The fact that its original designers were mainly developers of computer algebra systems lead to little attention being paid to geometry. The geometry-related Content Dictionaries one can find at the OpenMath website at the time of writing are bundled in a group called `plangeo`. There are six of them, `plangeo1,...,plangeo6`. The symbols defined in these CDs deal with planar Euclidean geometry and with generating polynomial systems from geometric configurations. These `plangeo` CDs were designed at Eindhoven University Of Technology as part of a project for automatically proving theorems sketched in the DGS Cinderella. Although Cinderella has an efficient randomized prover, its proofs cannot be verified. The goal of the project was to let Cinderella communicate with GAP by means of OpenMath, in order to use bracket algebra to obtain sound proofs of geometric theorems [3].

Another use of the `plangeo` CDs has been reported in [4]: constructions in Cabri, The Geometer's Sketchpad and Cinderella dealing with geometric loci, proving and discovering, are rewritten in OpenMath and exported to Mathematica and CoCoA for algebraic manipulation. As far as we know, no other use of the `plangeo` CDs has been published.

3 The Intergeo file format: design decisions

The file format of Intergeo is based on three major design decisions, which we explain below: the choice of zip-packaging, the choice of a constructions-based approach as opposed to a constraints-based approach, and the choice of OpenMath as semantic infrastructure. This paper provides a summary of the details described in [5].

3.1 Packaging

The Intergeo files, just as many other formats that have appeared recently, are ZIP archives containing several files. The most important file is the central file `intergeo.xml`; optional files can be compressed containing media and style elements which should be detached from the construction.

```
construction/
construction/intergeo.xml
construction/preview.svg
construction/preview.png
metadata/
metadata/i2g-lom.xml
resources/
resources/photo-jump.jpeg
```

The `intergeo.xml` file encodes the initial positions, in XML, and the constraints, in OpenMath being made of references to the Content Dictionaries. See [5] for a detailed specification of the archive. On the right you see an example archive listing.

An example content of `intergeo.xml` is given in figure 1 where one can see the usage of construction and constraint elements which we explain below.

3.2 Constructions-Based Description

The objective of the file format specification is providing a semantics of interactive geometry which should be understandable by all DGS implementors as well as further systems such as proof assistants. The mathematical semantics is, thus, important. In this section, we describe the chosen conceptual approach while the next describes the current concrete specification of OpenMath symbols that has been achieved.

Dynamic Geometry Systems deal with sets of geometrical objects that have certain relations. We call such a set of objects with given relations a *configuration.* All objects are part of some underlying space, for example the euclidean plane. In principle, if nothing else is said about them, objects can move around freely in this space. Relations then specify *constraints* on the movement of these objects.

Example 1. Two points P and Q, together with a line l; there is the following constraint:

$$\text{line } l \text{ is incident to both points } P \text{ and } Q.$$

Example 2. A circle Γ, a point P, a line l and the following constraints:

$$P \text{ is on } l$$
$$l \text{ is tangent to } \Gamma$$
$$\text{the distance of } P \text{ to the center of } \Gamma \text{ is } 10.$$

We can make the simple observation that the constraints do not determine the positions of the objects uniquely. This causes multiple problems that lie at the heart of dynamic geometry.

```
<construction>
  <elements>
    <point id="P">
      <homogeneous_coordinates>
        <double>2</double>
        <double>5</double>
        <double>1</double>
      </homogeneous_coordinates>
    </point>
    <line id="l">
      <homogeneous_coordinates>
        <double>7</double>
        <double>3</double>
        <double>-29</double>
      </homogeneous_coordinates>
    </line>
    <point id="Q">
      <homogeneous_coordinates>
        <double>5</double>
        <double>-2</double>
        <double>1</double>
      </homogeneous_coordinates>
    </point>
  </elements>
  <constraints>
    <line_through_two_points>
      <line out="true">l</line>
      <point>P</point>
      <point>Q</point>
    </line_through_two_points>
  </constraints>
</construction>
```

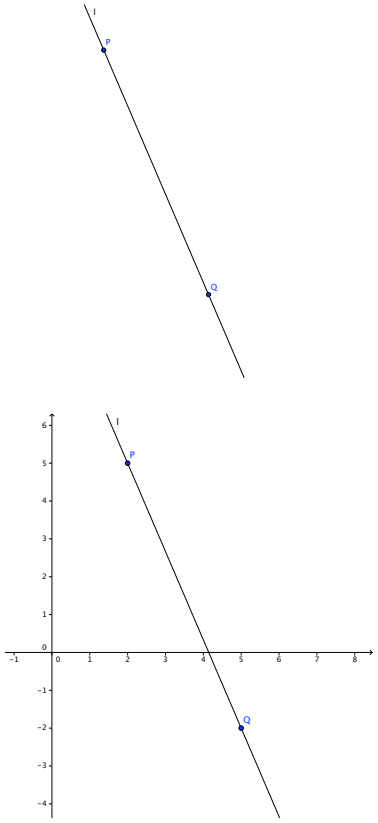

Figure 1: The content of an `intergeo.xml` for a simple construction of two points and one line through them. On the right, two possible (default) graphical representations of it: with or without axes; an important distinction between several interactive geometry systems which decide to present the geometry within a coordinate system or not per default

There is the problem of how to create an instance of the configuration. We say this has to do with the *static* aspect of the configuration. In Example 1, the points P and Q could still lie anywhere on the line l as it stands. For any instance, we must specify where l, P and Q should be. But once we have specified one, the other two are not completely free anymore. This particular example is not hard. And Example 2, although more difficult, is still doable. But in general, it is very difficult to give any particular solution for a set of constraints. There is not even a quick method to decide whether there are any instances: a set of constraints could be too restrictive and leave none.

Second, there is the dynamic behaviour of a configuration, caused by the freedom still left by the constraints. In Example 1, what should the user be able to move? May the line be picked up and translated or rotated in its entirety, the points being translated and rotated with it? Can the user only move one of the two points, the line being adjusted accordingly? Constraints of a strictly classical geometrical nature, such as the ones stated above, do not say anything about this behaviour. For the approach of a DGS, this is not enough.

A natural way to shed light on both these problems is a more precise specification of how the objects depend on each other. We could stipulate which objects are *free*, meaning that they can be varied over the whole range of possibilities in the underlying space (think of the plane) by the user. We would then proceed saying which objects depend only on the free objects, which ones depend only on these new objects and the free objects, etcetera. Such a specification is called a *construction*. It allows a DGS to rapidly create instances or decide that there are none. It also enables a DGS to give more consistent dynamic behaviour: objects are only movable insofar as they still have some degrees of freedom left, supposing the objects they depend on are kept fixed. The behaviour for all different cases (e.g. a line through a fixed point) can be decided in advance. Other objects dependent on the object being varied have to change as well, and this still leads to decision problems, but they are less severe. We could give a construction for Example 1 as follows:

Example 3. Two points P and Q, together with a line l, and the following construction:

$$\begin{gathered}\text{free\_point}(P)\\ \text{line\_through\_point}(l,P)\\ \text{point\_on\_line}(Q,l)\end{gathered}$$

The line l would then depend on where P is placed. That point could be varied freely. The line could then be rotated around P (and Q would most logically rotate with it), and while P and l are kept fixed, Q could still slide over the line. Note that such information could not be gleaned from the configuration.

It thus seems like a configuration might be too general to be practical, and we might be better off with a construction. We therefore decided to go with constructions. This decision implies less interoperability with constraint-based systems, since some of their resources will not be encodable into the format. But it ensures that construction-based DGSs will be able to interpret the resources,

which they might not if we used configurations. Indeed, although some systems like Geometer's Sketchpad [6] and Geometry Expressions [7] take a constraint-based approach, most systems use constructions.

Another effect of the decision is the potential explosion of keywords. We have to distinguish between "line_through_point" and "point_on_line". This is in sharp contrast to configurations, where one relation "incident" would suffice. In general, if there are n different types of objects, the construction approach now forces n^2 different types of incidence on us. This means a more bloated specification of the file format. On the other hand, it is easier for software developers to parse constructions, so it saves trouble there.

3.3 Design Decisions: OpenMath

The advantage of using OpenMath [8] as opposed to a self-chosen XML-format lies in the fact that the use of a Content Dictionary makes for a flexible, open, and reusable standard whose mathematical rules can be described. First of all, the use of OpenMath enables INTERGEO to use other Content Dictionaries already in existence, so it saves development time. Second, other kinds of software that want to use the format in the future can combine it with other Content Dictionaries to enrich its expressive power.

The choice of OpenMath also lies in the documentation of the extensibility: although the INTERGEO consortium groups a wide majority of the implementors of interactive geometry software and the agreement they have reached, the file formats of each of these softwares will evolve. Thanks to the formal-mathematical-properties, new symbols can be used by one software with a fair chance that they will be also usable, mathematically correctly, by other softwares [9].

The OpenMath XML syntax specified in [8] is not used within the file-format because of the numerous restrictions on the constraints and elements parts: they make it easy to use a more expressive XML syntax which uses the element-name instead of the symbol name. A look to the sample of figure 1 will allow the reader to easily translate. The format is used, however, within the content-dictionaries to express both the examples and the formal properties.

4 Content Dictionaries: achieved set of symbols

As the title of this paper indicates, the Intergeo File Format is still work in progress. The Intergeo team is regularly discussing issues and trying to reach consensus on a substantial part of them. At the end of July 2008, a first version had been constructed that pertained to lines and points. Although outdated now, it can still be found at `http://svn.activemath.org/intergeo/Deliverables/WP3/D3.3/`. At the end of July 2009, a second version was constructed, which looks more or less like described in this section. The content-dictionaries are edited after more informal writing happening in the project's wiki and a discussion has taken place. They can be browsed from `http://svn.activemath.org/intergeo/Drafts/Format/cd/`.

The final version will be available at the end of June 2010. This will also mean that at that time, all software partners will have a working API that supports this format. For more about APIs, see section 8.

The second version of the file format mainly concerns itself with points, lines, (directed) line segments and rays, polygons, and conics.

First, in the *elements part* of an i2g file, all geometric objects in the construction are declared. This implies that the valid OpenMath symbols appearing there, the vocabulary of the `intergeo_elements` Content Dictionary, is also the collection of mathematical *types* that we work with. This setup therefore makes explicit what kind of objects we consider to be first-class citizens in a geometric construction. For now, the types we have are:

Point	Line	Linear_equation
Direction	Ray	Line_segment
Directed_line_segment	Polygon	Conic
Ellipse	Circle	Parabola
Hyperbola	Locus	

We note that some types are subtypes of other types, e.g. circles are ellipses, which are themselves conics, just like parabolas and hyperbolas. Subtyping gives rise to some specific problems which we are tackling at the moment. For example, we are pondering including artificial types like `Linear_object` for generalizing `Line`, `Ray`, `Line_segment` and `Directed_line_segment`, as well as `Object` to encompass all types.

Once all objects in a construction have been declared in the *elements part* of the i2g file, in the *configuration part* the geometric relations between the objects can be described. The description is done by predicates. Formally, this means that a geometric relation is cast in the form of a functional type, in the sense of type theory, with input the types of the relevant objects and output type `Bool`. For example, we might say that a point is constrained to lie on a previously defined line by writing

$$\texttt{point\_on\_line}(p, l)$$

with type

$$\texttt{Point} \times \texttt{Line} \to \texttt{Bool}.$$

Some predicates do not refer to geometric relations but to dynamic behavior. For example, an object can be specified to be freely movable, independent of anything else. Since the behaviour of a free object may depend on its type, we chose to explicitly include this type, whence we have `free_point`, `free_line`, etc.

We currently have the following list of constraints:

free_point	free_line
point_on_line	line_through_point
line_through_two_points	line_perpendicular_to_line
line_perpendicular_to_line_through_point	line_parallel_to_line
line_parallel_to_line_through_point	point_intersection_of_lines
line_angular_bisector	line_segment_by_points
carrying_line_of_line_segment	endpoint_of_line_segment
point_on_line_segment	directed_line_segment_by_points
starting_point_of_directed_line_segment	end_point_of_directed_line_segment
line_segment_of_directed_line_segment	ray_from_point_to_point
ray_from_point_in_direction	starting_point_of_ray
carrying_line_of_ray	direction_of_ray
point_on_ray	point_on_conic

5 Basic Requirements: A Wish List

From our experience with the OpenMath geometric symbols (`plangeo` CDs), and mainly from the discussion with the software partners of the Intergeo project, we have described a basic comprehensive list of geometric elements and functions that should be supported by the common file format. Of course, it is not a complete and exhaustive list of all the elements from all DGSs involved but rather the set including the most common elements and functions. The purpose of this wish-list is hence to serve as a beacon to lead the way towards a common ground. It consists of a limited number of basic elements, but more elements can (and must) be added in the future.

The list has two parts: Constructions and Functions. In the *Constructions* part, with 53 elements, we list the basic elements of a geometric construction, from the simplest ones (a free point) to the more complicated ones (for example, a geometric locus). We have identified different categories of elements. For example, in the category *Points* we can find *Free point*, *Point on line*, ...

In the *Functions* part, with 10 elements, we consider calculations that can be performed on a geometric construction, like computing the area of a triangle or the length of a segment; or transformations on a construction, like a rotation.

The following is a schematic version of the list in which the number of construction subtypes has been added for each type.

- Constructions
  - Points (17), Vector (1), Segment (1), Lines (15), Rays (3), Polygons (2), Circles (6), Conics (6), Locus (2)
- Functions
  - Angle, Area, Length, Distance, Reflection wrt a point, Reflection wrt a line, Reflection wrt a circle, Rotation by two lines, Rotation by a point and an angle, Traslation

6 Implementation progress, library, and test-suite

A wide variety of DGSs exists nowadays. Before this project, each system used incompatible proprietary file formats to store its data. The Intergeo file format aims to be the convergence of the common features of the current DGSs together with the vision of future developments and the opinion of external experts. Its final version, based on modern technologies and planned to be extensible to capture the flavour of different DGSs, could serve as a standard in the DGS industry.

The specification of the first version of the Intergeo file format has been released as [5] after intensive collaboration between DGS software developers and experts. It specifies only a restricted subset of possible geometric elements, which however lead to an agreement on the structure and basic composition of the format.

As soon as Version 1 of the file format got more concrete, some of the software developers started to investigate its practical usage by integrating it (partially) into their software (see `http://i2geo.net/xwiki/bin/view/About/I2GformatImplementations`). It was possible to move simple content between several of the packages in the project. The gained experiences influenced the further steps: the development of the next version of the file format is ongoing, will be released before summer and can be followed on `http://svn.activemath.org/intergeo/Drafts/Format/cd/`. The development of a common API for the file format (ClientAPI) has started and will soon be available to all developers to simplify their implementation work.

The need of a test suite emerged to check the compatibility level of the different DGSs and to be able to release compatibility certificates. The test suite is composed of several test cases for the different elements and/or constraints. Its result is one of the following four levels:

- cannot read the element/constraint technically (which should not happen)
- can read the element/constraint technically -without throwing exceptions etc.- so it can at least be ignored (for unsupported elements/constraints)
- can read the element/constraint and represent it internally, though maybe with wrong or missing semantics (partial support of the element/constraint)
- can read the element/constraint and represent it internally with correct semantics (full support of the element/constraint)

Furthermore another API (ServerAPI) has been defined to allow the Intergeo platform to display DGS resources. It allows retrieving important information like a preview image, the HTML fragment and the MIME type. It will also enable converting the construction to the Intergeo file format. Thus all committed constructions will be available in the common file format as soon as the corresponding DGSs will implement both APIs.

7 Mathematical Issues and Problems

While developing the file format we identified several issues and mathematical problems that have to be addressed. We will discuss some of them briefly here; for further discussions we refer to the mailing list archives of Intergeo Work Package 3, and we also invite others to join the discussion there.[10]

This section repeats parts of the Deliverable 3.3, and highlights the important aspects for the OpenMath community. Despite the fact that we are listing a lot of yet to be solved issues, we are confident that the first version of the Intergeo file format is capable of handling any design decisions that result from the following discussion. For a description of some of the underlying mathematical problems, we refer to [10].

7.1 More Elements and Polymorphism

We already specified a wish list for further elements and constraints in Section 5, that contains the most important geometric extensions. Still, there are many other objects that have to be expressible by the file format, more general ones like functions or numbers, as well as stylistic elements like text objects or images. A task for the next version of the Intergeo file format is to collect and specify all elements that are currently in geometry software.

Certain elements are very similar to others, e.g. segments and rays can be used instead of lines in many cases. Other examples are arcs of circles in comparison to plain circles. The next version of the Intergeo file format has to be able to handle such polymorphism, as well as symbols that have a variable number of arguments. An example for the latter is the definition of polygons by vertices.

Some objects can also be replaced by others in certain special cases, for example, a circle might degenerate to a line, or a conic might degenerate to two lines. Although currently no DGS uses these degenerations, this could be desirable for the future. A DGS might construct a parallel line to a degenerate circle through three collinear points — with our current specification and typing mechanism it is not possible to capture this in the file format. However, we request advise from DGS developers and users on this issue.

7.2 Ambiguity Resolving

Ambiguity Resolving is crucial in finding the correct positions of elements in stored construction after loading, also known as the persistent naming problem [11] from parametric CAD. Assume that an intersection point of two circles is used in a construction. If the two circles are moved into a tangent position, and then the construction is stored, then both intersections have the same coordinates and thus cannot be distinguished. If the circles are moved into a position where both intersection points can be distinguished, then it is essential to pick the correct intersection point.

[10] See `http://lists.inter2geo.eu/mailman/listinfo/wp3`

Most DGSs solve this problem by having an implicit order of multiple outputs. This order is dependent on the implementation details of the algorithms and cannot be part of a specification. Also, a point might switch branches later due to homotopy-conserving implementations. A detailed review of these problems can be found in [12]. This means that this approach cannot be used for a cross-software file format.

As soon as circle (or conic) intersections are introduced, we will have to find a solution to this problem.

7.3 Functions and Scripting

Almost all DGSs support some form of plotting graphs, defining element dependencies, or changing the style of elements dynamically by using functions that are defined symbolically. All of these use a different language to specify the functions, though many aspects are shared. The conformance to standards varies wildly between "OpenMath compliance" and "unspecified."

Right now it seems impossible to homogenize the various dialects. Actually, the translation from one language to the other can be done easily by humans if an automatic conversion fails, so we decided that for now all functions should be specified in the private sections of the file format. Each DGS may try to interpret the other function specification, of course, and store its own interpretation as well.

For this, we need a notion of "alternatives", which will be specified in an upcoming version of the Intergeo file format.

Another difficulty in dealing with functions is that some DGSs extend the notion of function to a general-purpose functional programming language. This proves that it is impossible to find equivalent functions algorithmically. Nevertheless, in many cases the translation is straightforward, and so it might be sufficient to use a heuristic approach.

One solution would be to use the API approach as described in Section 8. Actually, a DGS A could use the "function dialect" of another DGS B by asking the other system B to interpret the native parts that A cannot understand via the API. While this sounds very entangled, it may be the general solution to the specification problem we face here.

7.4 Mathematical Typesetting

Usually, mathematical typesetting is done with TeX [13], and a browser-compliant way is to use MathML [14]. DGS software uses both approaches, while the TeX implementation used is usually only a subset of the full TeX system as created by Knuth[11].

We could not agree on a definitive way to typeset formulae. Probably it would be a good idea to support MathML, but most of the DGS developers do not want to adopt it, as it seems. So this is currently unspecified and mathematical typesetting has to be specified in the private part of a construction.

[11] Some use the hoteqn library, others use custom implementations

7.5 Macro Constructions

So far there is no notion of *macro constructions* in the file format specification. We expect to treat them basically as subconstructions, and it is probably sufficient to add an additional `inmacro` attribute to the constraints and elements. However, we have postponed this matter until we have a more substantial set of examples.

7.6 Number Representation

Currently, the specification of coordinates uses the IEEE standard for doubles for historical reasons. While this is probably sufficient for most purposes, it lacks the ability to describe *real* coordinates, for example the irrational numbers π or $\sqrt{5}$. As there are constructions even in elementary geometry that require such numbers, it is desirable to be able to express them.

The OpenMath standard provides a means to specify all the real numbers likely to appear in normal applications of Interactive Geometry, but some implementation difficulties have been pointed out. For the time being, we will restrict the number representation to the IEEE standard, in particular due to the reason that no DGS so far uses another specification. This should not be the cause of severe problems, because the coordinates of dependent elements can be recalculated up to arbitrary precision by the DGS itself. If there is a need for other number representations, we will extend the mechanism. This will not need a major redesign, as all occurrences of numbers will just be replaced with real number values.

8 File Format and API Interaction

The obvious way to use the file format is to exchange files; save them in one DGS and load them in the other. While this is usually done manually, the file format is even more useful when it comes to automated exchange between software. Therefore, we are currently defining an application programming interface (API) that will make use of the file format at several places [15].

Basic interaction facilities of the API are loading and saving of files, both in native and Intergeo file format, which allows for creating server-side software that delivers files in the users' preferred format. Also, copy and paste operations will be described in terms of the Intergeo file format, as any copied part of a construction is in itself a construction.

We also allow for communication from and to non-DGS software, e.g. a CAS. Getting and setting values like objects' coordinates using a CAS opens a wide variety of applications.[12] In particular, most DGSs currently work with double

[12] As a reviewer points out it is a problem to send values like π to a CAS if there is no internal representation for it in the DGS. In practise it usually does not pose a problem, as only coordinates of free elements have to be sent, and there is no need to set a free element to exact real coordinates.

precision internally, which is not sufficient for research-grade use. As a primary concern of interactivity is real-time operation, it is not surprising that most software only uses a precision that can be handled in hardware. Using the API it is possible to externalize this calculations in situations where we can sacrifice time for exactness.

Many DGSs already use built-in facilities for algebraic (or better: analytic, as the focus is on doing calculations) manipulation. This is a major obstacle for compatibility, as all the dialects are different. Using the API it could be possible to use the DGS part of one software package, while using the CAS part of the other. Again, the core of the interaction is the clear specification of Intergeo elements.

Finally, we want to mention the pedagogical benefits of a standardized API. The integration of interactive exercises that can be checked by other software (like theorem proving systems) could be a huge step for technology-based teaching and learning of mathematics.

9 Outlook

This paper has presented the current progress on the development of the Intergeo file format towards interoperable interactive geometry: the choices made and the implementations achieved. We believe that the format will support sharing helped by the Intergeo platform which aims at breaking the other barrier of sharing interactive geometry: the barriers between educational regions.

As a more current preoccupation, the team of authors is involved into the maturation process with a strong hope to have a satisfactory format to indeed break many software barriers that split interactive geometry communities in too small chunks. We expect that it will renew a competition among dynamic geometry systems' shining by their exclusive features but providing the basic exchange format for the core of interactive geometry.

The common file format has attracted several interested parties thus far, all makers and users of softwares at the edge of interactive geometry, in such domains as algebraic geometry.

References

1. Kortenkamp, U., Blessing, A.M., Dohrmann, C., Kreis, Y., Libbrecht, P., Mercat, C.: Interoperable interactive geometry for europe – first technological and educational results and future challenges of the intergeo project. In: Proceedings of CERME 6, Lyon. (2009)
2. Kortenkamp, U., Dohrmann, C., Kreis, Y., Dording, C., Libbrecht, P., Mercat, C.: Using the intergeo platform for teaching and research. In Bardini, C., Fortin, C., Oldknow, A., Vagost, D., eds.: Proceedings of the 9th International Conference on Technology in Mathematics Teaching, Metz, ICTMT-9 (2009)
3. Roozemond, D.: Automated proofs using bracket algebra with cinderella and openmath. In: RWCA 2004: Proceedings of the 9th Rhine Workshop on Computer Algebra. (2004)

4. Escribano, J., Botana, F., Abánades, M.: Adding remote computational capabilities to dynamic geometry systems. Mathematics and Computers in Simulation (To appear)
5. Grothmann, R., Hendriks, M., Kortenkamp, U., Kreis, Y., Laborde, J.M., Libbrecht, P., Marquès, D.: D3.3: Common file format v1. Technical report, Intergeo Project (2008) See `http://i2geo.net/files/D3.3-Common-File-Format-v1.pdf`.
6. KCP Technologies Inc.: Geometer's sketchpad v4 (2008)
7. Saltire Software: Geometry expressions v1.1 (2008)
8. Stephen Buswell, Olga Caprotti, D.C.M.D.M.G., Kohlhase, M.: The OpenMath Standard, version 2.0. Technical report, The OpenMath Society (2004) Available at `http://www.openmath.org/`.
9. Davenport, J.H., Libbrecht, P.: The freedom to extend openmath and its utility. Journal of Computer Science and Mathematics **59** (2008) 1–19 NL=yes;RB=Jan.
10. Kortenkamp, U.: Foundations of Dynamic Geometry. Dissertation, ETH Zürich, Institut für Theoretische Informatik, Zürich (1999)
11. Marcheix, D., Pierra, G.: A survey of the persistent naming problem. In: SMA '02: Proceedings of the seventh ACM symposium on Solid modeling and applications, New York, NY, USA, ACM (2002) 13–22
12. Denner-Broser, B.: Tracing-Problems in Dynamic Geometry. PhD thesis, Freie Universität Berlin, Berlin (2008)
13. Knuth, D.E.: The TEXbook. Addison-Wesley, Reading, Massachusetts (1984)
14. Carlisle, D., Ion, P., Miner, R.: Mathematical markup language (mathML) version 3.0. World Wide Web Consortium, Working Draft WD-MathML3-20080409 (2008)
15. Kortenkamp, U., Kreis, Y.: D3.5: I2g api specification. Technical report, Intergeo Project (2009) To appear. See `http://i2geo.net/files/D3.5-API-specification.pdf`.

Semantics of OpenMath and MathML3

Michael Kohlhase, Florian Rabe

Computer Science, Jacobs University Bremen
{m.kohlhase,f.rabe}@jacobs-university.de

Abstract. Even though OpenMath has been around for more than 10 years, there is still confusion about the "semantics of OpenMath". As the upcoming MathML3 recommendation will semantically base Content MathML on OpenMath Objects, this question becomes more pressing.
One source of confusions about OpenMath semantics is that it is given on two levels: a very weak algebraic semantics for expression trees, which is extended by considering mathematical properties in content dictionaries that interpret the meaning of (constant) symbols. While this two-leveled way to interpret objects is well-understood in logic, it has not been spelt out rigorously for OpenMath.
In this paper we look at the semantics of OpenMath from a foundational point of view and reconcile this "semantics" with the foundations of mathematics established in the early 20th century; the traditional way of assigning meaning to mathematical objects.

1 Introduction

MathML2 [ABC+03] and OpenMath2 [BCC+04] are standards for the representation and communication of mathematical objects. Even though they have been around for more than 10 years, there is still confusion about the "semantics of OpenMath". As the upcoming MathML3 recommendation will semantically base Content MathML on OpenMath Objects, this question becomes more pressing.

1.1 OpenMath and MathML

MathML comes in two parts: *presentation* MathML, which provides XML-based layout primitives for the traditional two-dimensional notation of mathematical formulae and *content* MathML, which focuses on encoding the meaning of objects rather than visual representations to allow the free exchange of mathematical objects between software systems and human beings. OpenMath has the same goals as content MathML, but was developed by a different community with slightly different intuitions. Both representation formats represent mathematical objects as expression trees. Content MathML tries to cover all of school and engineering mathematics (the "K-14" fragment) in a representation format intuitive to mathematicians, and OpenMath concentrates on an extensible framework built on a minimal structural core language with a well-defined

extension mechanism. Where MathML supplies more than a dozen elements for special constructions, OpenMath only supplies concepts for function application (`OMA`), binding constructions (`OMBIND`), and attributions (`OMATTR`). Where MathML provides close to 100 elements for the K-14 fragment, OpenMath gets by with only an `OMS` element that identifies symbols by pointing to declarations in an open-ended set of Content Dictionaries.

An OpenMath Content Dictionary (CD) is a document that declares names (OpenMath "symbols") for basic mathematical concepts and objects. CDs act as the unique points of reference for OpenMath symbols (via `OMS` elements) and thus supply a notion of context that situates and disambiguates OpenMath expression trees. To maximize modularity and reuse, a CD typically contains a relatively small collection of definitions for closely related concepts. The OpenMath Society maintains a large set of public CDs [OMC08], including CDs for all pre-defined symbols in MathML2. There is a process for contributing privately developed CDs to the OpenMath Society repository to facilitate discovery and reuse. OpenMath does not require CDs be publicly available, though in most situations the goals of semantic markup will be best served by referencing public CDs available to all user agents.

To avoid fragmentation and to smoothe out interoperability obstacles, effort is currently under way to align OpenMath and MathML semantically. To remedy the lack of regularity and specified meaning in MathML, content MathML was extended by concepts like binding structures and full semantic annotations from OpenMath and a structurally regular subset of the extended content MathML was identified that is isomorphic to OpenMath objects. This subset is called **strict content MathML** to contrast it to full content MathML that is seen to strike a more pragmatic balance between regularity and human readability. Full content MathML borrows the semantics from strict MathML by a mapping specified in the MathML3 specification [ABC+09] that defines the meaning of non-strict (**pragmatic**) MathML expressions in terms of strict MathML equivalents. Strict Content MathML in turn obtains its meaning by being an encoding of OpenMath Objects.

In this situation, the "meaning of OpenMath (Objects)" obtains a completely new significance; especially when OpenMath still receives evaluations like

> *On the other hand the paper leaves me unsatisfied, and even irritated. It is frustrating to know that the* MathML*3 and* OpenMath*3 standards still will be meaningless from a semantic point of view. [...] will not lead to a standard for mathematical expressions where those expressions have a proper semantics.* anonymous referee for [DK09]

The aim of this paper is to clarify the status of semantics in OpenMath (and thus content MathML3) and in particular counter sentiments like the one above. We see the reason for this "misunderstanding" in a presentational gap between how mathematical objects and theories are conventionally given a meaning and the way OpenMath answers the question. In the rest of this section, we will briefly recap the established foundations (of meaning in) Mathematics and the

way OpenMath establishes meaning. Based on this, we will bridge the differences and clarify gray areas in a formal semantic analysis: in Section 2 we develop an algebraic semantics for OpenMath objects and in Section 3 we extend this to a model-theoretic semantics for OpenMath objects. Section 4 concludes the paper.

1.2 Foundations of Mathematics

The question of what the meaning of mathematical expressions and theories might be is usually by methods from **Logic**, a scientific field at the intersection of philosophy and mathematics concerned with the study of the concepts **proposition** and **truth** and the reasoning about them.

The age-old question about the meaning of language in general and mathematics in particular turned into the "*Grundlagenkrise*" of mathematics by the discovery of paradoxa, i.e., contradictions, in what is called **naive set theory** in retrospect. Naive set theory was the implicitly assumed foundation of mathematics at the time, Cantor's "Grundlagen" [Can83] from 1883 being the most influential contribution. The best known paradoxon was found by Bertrang Russell in 1901 [Rus01]. Giuseppe Peano had noticed a similar one in 1897.

In response to this, mathematicians have developed several — sometimes alternative, sometimes complementary — foundations (i.e. specific logics picked as a starting point of mathematics) that can replace naive set theory. This happened over several decades as an evolutionary creative process. But it did not culminate in a commonly accepted solution. Rather, it led to profound and sometimes fierce debates on what mathematics is. The personal quarrel between Hilbert and Brouwer, which was partially fuelled by these debates, is an almost tragic example. From this evolution emerged two major classes of foundations: **axiomatic set theory** and **type theory**.

The basic idea of **axiomatic set theory** is that there is a universe of sets, and any mathematical object ever introduced is a set. The sets are related via the binary relations of equality and membership. For example $m \in M$ is used to say that the set m is a member of the set M. Depending on context, M is regarded as a property of m or as a structuring concept. To talk about sets, equality, membership, and propositions are used. The basic propositions are of the form $m = m'$ and $m \in M$. Composed propositions are built up from the basic ones. Typically, (at least) first-order logic (FOL) is used as the language of composed propositions: FOL uses propositions such as $F \wedge G$ and $\forall x.F(x)$ denoting "F and G are true" and "for all (sets) x, F is true about x". Then a limited collection of propositions (the **axioms**) is chosen as fundamental truths. These are chosen very carefully to prevent contradictions and to obtain a minimal set of axioms. Based on the axioms, **proofs** are used to single out the true propositions. A proof consists of a sequence of steps that derive one true proposition from other true propositions starting with the axioms. In this way the whole of mathematics is developed, and for every proposition, truth is defined by whether it has a proof.

Both set theory and type theory have led to **numerous specific foundations** of mathematics. Zermelo-Fraenkel set theory, based on [Zer08,Fra22], is

most commonly in use today. Other variants are von Neumann-Bernays-Gödel set theory, based on [vN25,Ber37,Göd40], which is important for category theory, and Tarski-Groethendieck set theory, based on [Tar38,Bou64]. The first type theory was Russells's ramified theory of types [Rus08]. And in their Principia [WR13], Whitehead and Russell gave one of the most influential foundations of mathematics. Church's simple theory of types, also called higher-order logic, [Chu40] is the most-used type theory today. Important other type theories are typically organized in the lambda cube [Bar91] and include dependent type theory [ML74,HHP93], System F [Gir71,Rey74], and the calculus of constructions [CH88]. Most of these foundations have further variants, such as Zermelo-Fraenkel set theory with or without the Axiom of Choice or type theory with or without product types.

Hilbert's formalistic program, set forth in his second problem [Hil00] and various texts from the 1920s, e.g., [Hil26], called for the reduction of all mathematics to a set of axioms and a consistency proof for these axioms using only finitary means. Since proofs are built up from the axioms, such a reduction would yield all true propositions by systematically searching all proofs. In 1930, Gödel established two negative results [Göd31], which as von Neumann recognized first showed that the goal of Hilbert's program is unreachable.

Gödel's first result roughly says that no foundation of mathematics can be found that defines the truth of all propositions in an algorithmic way. The second one says that no foundation can prove its own consistency. Gödel worked in the Principia — the foundation mainly in use at the time — but the results extend to all foundations beyond a certain level of expressivity. This is the major reason why no foundation has won the endorsement of mathematicians as a whole and why there will not be a final answer which foundation of mathematics is the best. Since no perfect foundation exists, the personal preferences and the characteristics of a problem lead to different choices of foundation.

In any case, most mathematicians today accept a mathematical object or theory as "meaningful", iff it can (in principle) be formalized in one of the foundations, most notably set theory. Incidentally, most logics (and type theories) in use have a set-theoretic (i.e. foundational) semantics that makes them acceptable to mathematicians in this sense.

1.3 The Meaning of OpenMath

The OpenMath standard actually gives two answers to the question about the meaning of OpenMath expressions. The first one comes from the fact that OpenMath is intended as a communication standard between mathematical software systems: OpenMath envisions communication via *phrasebooks* ([AvLS98] or see [BCC+04, chapter 1]): Each mathematical software system S is equipped with an OpenMath phrasebook that converts OpenMath expressions from and to the internal representations of the system S. In this "system communication view", the meaning of OpenMath expressions is built into the phrasebooks that (purport to) understand the expression, and the meaning is whatever S (after conversion by the phrasebook) makes it to be. Clearly, this

view of meaning is not very helpful, and taken in the radical simplicity we have formulated it here is not an adequate account. After all, the purpose of the OpenMath standard is to synchronize the system-specific representations of objects, so that communication between systems is meaning-preserving. To attain this goal, OpenMath does two things:

1. It defines the class of "OpenMath objects" which acts as the model for encodings of mathematical formulae. OpenMath objects are essentially labeled trees modulo α-conversion for binding structures and flattening for nested semantic annotations. The OpenMath standard considers OpenMath objects as primary citizens and views the "OpenMath XML encoding" as just an incidental design choice for an XML-based markup language. In fact OpenMath specifies another encoding: the "binary encoding" designed to be more space efficient at the cost of being less human-readable.
2. Rather than appealing to mathematical intuition, OpenMath stipulates that phrasebooks should be informed by (mathematical properties in) content dictionaries.

   > *It is the OpenMath Content Dictionaries which actually hold the meanings of the objects being transmitted. For example if application A is talking to application B, and sends, say, an equation involving multiplication of matrices, then A and B must agree on what a matrix is, and on what matrix multiplication is, and even on what constitutes an equation. All this information is held within some Content Dictionaries which both applications agree upon. [...] The primary use of Content Dictionaries is thought to be for designers of Phrasebooks, the programs which translate between the OpenMath mathematical object and the corresponding (often internal) structure of the particular application in question.* [BCC+04, section 4.1]

Even if this is not spelt out[1] in the OpenMath2 standard the algebra $\mathcal{O}$ of OpenMath objects can be interpreted[2] as an (initial) model for encodings of mathematical formulae. Note that since $\mathcal{O}$ is initial it is essentially unique and identifies (in the sense of "declares to be the same") fewer objects than any other model. As a consequence two mathematical objects must be identical if their OpenMath representations are, but not the other way around.

While this can be seen as a failure of OpenMath to supply semantics ("OpenMath *is only syntax"*), we see it as an expression of the OpenMath representational philosophy expressed in

> *OpenMath objects do not specify any computational behaviour, they merely represent mathematical expressions. Part of the OpenMath philosophy is to leave it to the application to decide what it does with an object once*

[1] In particular the "compliance chapter" does not mention mathematical properties in CDs at all.

[2] To the best of our knowledge, this "act of interpretation" has never been backed by a formal mathematical study; which is what prompted the work reported in this paper.

> *it has received it. OpenMath is not a query or programming language. Because of this, OpenMath does not prescribe a way of forcing "evaluation" or "simplification" of objects like* $2+3$ *or* $\sin(\pi)$*. Thus, the same object* $2+3$ *could be transformed to* 5 *by a computer algebra system, or displayed as* $2+3$ *by a typesetting tool.* [BCC+04, section 1.5]

In this sense the initial algebra semantics of OpenMath objects is intentionally weak to make the OpenMath format ontologically unconstrained and thus universally applicable. It basically represents the accepted design choice of representing objects as formulae. Any further (meaning-giving) properties of an object o are relegated to the content dictionaries referenced in o, where they can be specified formally (as "Formal Mathematical Properties" in `FMP` elements containing XML-encoded OpenMath objects) or informally (as "Commented Mathematical Properties" in `CMP` elements containing text). Thus the precision of OpenMath as a representation language can be adapted by supplying CDs to range from fully formal (by providing CDs based on some logical system) to fully informal (where CDs are essentially empty except for declaring symbols).

In the next section, we will formally develop the initial algebra semantics of OpenMath objects, and then in section 3 extend it to take mathematical properties in CDs into account, thus showing that the interpretation above can indeed be made mathematical and be reconciled with the notion of meaning in foundations of mathematics.

2 An Algebraic Semantics for OpenMath Objects

We will now define a an algebraic semantic semantics for OpenMath objects building on ideas from [BBK04]. The difference to the situation there (giving a semantics for the simply typed λ calculus with a type of Booleans) is that OpenMath allows n-ary function application (rather than binary) arbitrary binding symbols (rather than just λ-abstraction), and arbitrary attributions (rather than just simple types), but only assumes α-conversion (rather than $\alpha\beta\eta$ conversion).

2.1 Syntax

We start out by fixing an abstract syntax of "OM objects", which we will relate to OpenMath objects in Section 2.3. We will call the objects specified in Definition 4 "*abstract* OM Objects" when we want to distinguish from the "*standard* OpenMath objects" defined in the OpenMath2 standard [BCC+04, section 2].

Definition 1 (Symbols and Variables). In all of the following, we will assume the existence of two disjoint, countably infinite sets: a set *Symbols* of **symbols** and a set *Variables* of **variables**. Furthermore, we assume a set $Keys \subseteq Symbols$ of **keys**.

As usual in formal languages we are a little more careful about the variables we use in the construction of complex objects. The notions of vocabularies and contexts help us do this.

Definition 2 (OM Vocabulary). An OM **vocabulary** is a set of symbols. For every OM vocabulary T, we denote by $\mathit{Symbols}(T) := \mathit{Symbols} \cap T$ the **set of symbols of** T and by $\mathit{Keys}(T) := \mathit{Keys} \cap T$ the **set of keys of** T.

Definition 3 (OM Context). An OM **context** C is an n-tuple of variables which we will write as $\langle x_1, \ldots, x_n \rangle$. We will use $+$ for tuple concatenation and $\in$ for tuple membership.

Definition 4 (OM Objects). Let T be an OM vocabulary. The set $O(T, C)$ of **OM objects** over T in context C is the smallest set closed under the following operations

1. if $s \in \mathit{Symbols}(T) \setminus \mathit{Keys}(T)$, then $\mathbb{S}(s) \in O(T, C)$,
2. if $x \in C$, then $\mathbb{V}(x) \in O(T, C)$,
3. if $f, o_1, \ldots, o_n \in O(T, C)$, then $\mathbb{A}(f, o_1, \ldots, o_n) \in O(T, C)$,
4. if $b \in O(T, C)$, $X_1, \ldots, X_n \in \mathit{AttVar}(T, C)$, and $o \in O(T, C')$ where $C' = C + \langle \mathit{varname}(X_1), \ldots, \mathit{varname}(X_{l(\sigma)}) \rangle$, then $\mathbb{B}(b, [X_1, \ldots, X_n], o) \in O(T, C)$,
5. if $o \in O(T, C)$, $k \in \mathit{Keys}(T)$, and $v \in O(T, C)$, then $\mathbb{K}(o|k := v) \in O(T, C)$.

Here **attributed variables** are defined by: $o \in \mathit{AttVar}(T, C)$ if a $o = \mathbb{V}(x)$ for some $x \in C$ or $o = \mathbb{K}(o'|k := v) \in O(T, C)$ for some $o' \in \mathit{AttVar}(T, C)$. We call OM objects in the empty context **ground**. The name of an attributed variable is defined by $\mathit{varname}(\mathbb{K}(o'|k := v)) = \mathit{varname}(o')$ and $\mathit{varname}(\mathbb{V}(x)) = x$.

Note that in contrast to the OpenMath2 standard we only consider "unary" attributions that associate an object with a single key/value pair. This allows us to build the "flattening of attributions" into the abstract representation of OM Objects. We can regain the syntactic structure of OpenMath2 objects by introducing n-ary attributions as an abbreviation for nested attributions: $\mathbb{K}(o|k_1 := v_1, \ldots, k_n := v_n) = \mathbb{K}(\mathbb{K}(o|k_1 := v_1)|k_2 := v_2, \ldots, k_n := v_n)$ for $n \geq 2$. With this trick[3] we have fully covered the requirement of "attribution flattening equivalence" required in the OpenMath standard.

Let us fortify our intuition with an example which will use throughout the paper; we represent binding objects, since they are the problematic cases.

Example 1. The untyped universal quantification $\forall x.x = x$ is represented as $\mathbf{U} = \mathbb{B}(\mathbb{S}(\forall), [\mathbb{V}(x)], \boxed{x = x})$[4], where $\forall$ is a symbol. To show the interaction of attribution and binding, we use a typed identity function represented as a λ-abstraction: $\lambda x : \beta.x$ is represented as $\mathbf{L} = \mathbb{B}(\mathbb{S}(\lambda), [\mathbb{K}(\mathbb{V}(x) \,|\, \tau := \boxed{\beta})], \mathbb{V}(x))$, where τ is a key symbol (i.e. a symbol with role "semantic-attribution"). We have $\mathbf{U} \in O(\{\forall, =\}, \langle\rangle)$ and $\mathbf{L} \in O(\{\lambda, \tau, \beta\}, \langle\rangle)$

[3] In fact we propose to follow this path in the next version of the OpenMath standard as it simplifies the presentation. Note that we are only talking about (standard) OpenMath objects, not their XML or binary encodings, where n-ary attributions make sense for notational convenience.

[4] Here and throughout the paper we will use boxed mathematical formulae to gloss OpenMath objects (encoded, abstract, or standard; we assume that this distinction is either meaningless or clear from the context).

The use of attributed variables in binders can lead to a somewhat awkward notations when accessing the keys and attributions present in abstract binding objects. Therefore, we use the auxiliary definition of binding signatures in the technical developments below. Intuitively, an OM binding object has binding signature σ if it binds $l(\sigma)$ variables where the i-th variable has $d^i(\sigma)$ attributions.

Definition 5 (Binding Signature). A **binding signature** σ consists of

- a positive natural number $l(\sigma)$ (the **length** of σ),
- natural numbers $d^1(\sigma), \ldots, d^n(\sigma)$ (the **depth of σ at** i).

We denote by $\overline{\sigma}$ the set of pairs $\langle i, j\rangle \in \mathbb{N} \times \mathbb{N}$ where $1 \leq i \leq l(\sigma)$ and $1 \leq j \leq d^i(\sigma)$. If σ is a binding signature with length n, $b \in O(T, C)$, $K : \overline{\sigma} \to Keys(T)$, and $V : \overline{\sigma} \to O(T, C)$, and $o \in O(T, C + \langle x_1, \ldots, x_n\rangle)$, then we write

$$\mathbb{B}(b\,[x_1, \ldots, x_n | K := V].o) \quad \text{for} \quad \mathbb{B}(b, [X_1, \ldots, X_n], o) \in O(T, C)$$

where $X_i = \mathbb{K}(\mathbb{V}(x_i)|K(i,1) := V(i,1), \ldots, K(i, d^i(\sigma)) := V(i, d^i(\sigma)))$.

Example 2 (Continuing Example 1). In the abbreviated syntax $\forall x.x = x$ is represented as $\mathbf{U} := \mathbb{B}(\mathbb{S}(\forall)\,[x].\boxed{x = x})$ and $\lambda x : \beta.x = x$ as $\mathbf{L} := \mathbb{B}(\mathbb{S}(\lambda)\,[x|K := V].\mathbb{V}(x))$, where

- $l(\sigma) = 1$ and $d^1(\sigma) = 1$, and therefore $\overline{\sigma} = \{\langle 1, 1\rangle\}$
- $K = \{\langle 1, 1\rangle \mapsto \tau\}$ and $V = \{\langle 1, 1\rangle \mapsto \boxed{\beta}\}$

Clearly, every OM object of the form $\mathbb{B}(b, [X_1, \ldots, X_n], o)$ can be written uniquely as an expression of the form $\mathbb{B}(b\,[x_1, \ldots, x_n | K := V].o)$, and we will use the latter notation in the future and abbreviate $\mathbb{B}(b\,[x_1, \ldots, x_n | \varnothing := \varnothing].o)$ with $\mathbb{B}(b\,[x_1, \ldots, x_n].o)$.

Definition 6 (Substitution). For $o \in O(T, \langle x_1, \ldots, x_n\rangle)$, the substitution function that maps $\langle o_1, \ldots, o_n\rangle$ to the object with all variables x_i substituted with o_i is denoted by $Subs(o)$.

Definition 7 (α-Equality). Two objects are said to be **α-equal** iff they arise from one another by renaming bound variables. $\equiv_\alpha$ denotes the induced equivalence relation, and $[o]_\alpha$ denotes the equivalence class of o.

2.2 Semantics

In the following, we will use use the notation $\Lambda x \in A.f(x).$ for the set-theoretical function defined by $\{\langle x, f(x)\rangle : x \in A\}$. A may be omitted if it is clear from the context. We also write B^A for the set of functions from A to B.

Definition 8 (OM Algebra). Let T be an OM vocabulary. An **OM algebra** A over T consists of

1. a set $U := U^A$ called the **universe of discourse**
2. a family of sets $R^A_n \subseteq U^{(U^n)}$ for $n \geq 1$,
3. an element $s^A \in U$ for every $s \in \mathit{Symbols}(T) \setminus \mathit{Keys}(T)$,
4. a family of mappings $@^A_n : U \times U^n \to U$ for $n \geq 1$,
5. a family of mappings $\beta^A_K : U \times U^{\overline{\sigma}} \times R_{l(\sigma)} \to U$ for mappings $K : \overline{\sigma} \to \mathit{Keys}(T)$ and binding signatures σ,
6. a family of mappings $\alpha^A_k : U \times U \to U$ for every $k \in \mathit{Keys}(T)$.

Whereas s^A, $@^A$, β^A, and α^A are intended to interpret symbols, applications, bindings, and attributions, respectively in a relatively standard fashion, the sets R^A_n are special. Because OpenMath permits arbitrary expressions as binders, it is not possible to define the interpretation of every binder separately as is common in both first-order and higher-order settings. Instead, we need to model variable binding explicitly in the semantics. Syntactically, binders are operators that take terms with free variables as arguments. It is well-understood in higher-order logic and type theory that terms with n free variables can be modeled as n-ary functions on the universe. Thus, we interpret binders as operators taking functions as arguments. These come from the $R^A_{l(\sigma)}$ in the third argument of β operator (note that $l(\sigma) = n$ here). The functions from $\overline{\sigma}$ to U^A in the second argument are used for dealing with the keys of the attributed variables.

Since we can always write a binder like $\mathbb{B}(b\,[x].\mathbb{V}(x))$ (for the empty binding signature), the R^A_1 should at least contain the identity function. However, the whole set $U^{(U^n)}$ is too big since only some of these functions actually arise from the interpretation of terms with free variables. Since the interpretation of these terms depends on A itself, we permit an arbitrary set R^A_n here and leave it to Def. 10 to sort out when an OM algebra is well-defined.

Definition 9 (Assignment). Let A be an OM Algebra over T, and let C be an OM context. An A-**assignment** φ for C is a mapping from C to U^A. We denote by $\varphi, [x/o]$ the assignment for $C + \langle x \rangle$ that maps x to o and agrees with φ for all other variables.

Definition 10 (Interpretation). Let A be an OM Algebra over T, and let φ be an A-assignment for a context C. The **interpretation** $[\![o]\!]^A_\varphi$ of $o \in O(T, C)$ in A under φ is defined as follows:

1. $[\![\mathbb{S}(s)]\!]^A_\varphi = s^A$,
2. $[\![\mathbb{V}(x)]\!]^A_\varphi = \varphi(x)$,
3. $[\![\mathbb{A}(f, o_1, \ldots, o_n)]\!]^A_\varphi = @^A_n([\![f]\!]^A_\varphi, \langle [\![o_1]\!]^A_\varphi, \ldots, [\![o_n]\!]^A_\varphi \rangle)$,
4. $[\![\mathbb{B}(b\,[x_1, \ldots, x_n | K := V].o)]\!]^A_\varphi = \beta^A_K([\![b]\!]^A_\varphi, \mathcal{V}, \mathcal{F})$ where
   (a) σ is the binding signature of the binding (which must have length n),
   (b) $\mathcal{V} = \Lambda p \in \overline{\sigma}.[\![V(p)]\!]^A_\varphi$,
   (c) $\mathcal{F} = \Lambda u \in (U^A)^n.[\![o]\!]^A_{\varphi,[x_1/u_1,\ldots,x_n/u_n]}$
5. $[\![\mathbb{K}(o | k := v)]\!]^A_\varphi = \alpha^A_k(o, v)$.

Whether the case for bindings is well-defined, depends on the sets R^A_n. We call A **well-defined** if $\Lambda u \in (U^A)^n.[\![o]\!]^A_{\varphi,[x_1/u_1,\ldots,x_n/u_n]} \in R^A_n$ for all C, n, $o \in O(T, C)$, and φ.

Example 3 (Continuing Example 2). To interpret **U** we use an OM Algebra A with

1. $U^A := \mathbb{N} \cup \{\mathsf{q}, \mathsf{e}, \mathsf{t}, \mathsf{f}\}$
2. $R^A_n = U^{(U^n)}$,
3. $\forall^A := \mathsf{q}$ and $=^A := \mathsf{e}$,
4. $@^A_2(\mathsf{e}, u, v) = \mathsf{t}$ if $u = v$ and $@^A_n(u, \langle u_1, \ldots, u_n\rangle) = \mathsf{f}$ otherwise.
5. $\beta^A_\varnothing(\mathsf{q}, \varnothing, \mathcal{F}) = \mathsf{t}$ if $\mathcal{F}(u) = \mathsf{t}$ for all $u \in \mathbb{N}$; and $\beta^A_K(u, \langle x_1, \ldots, x_n\rangle, \mathcal{F}) = \mathsf{f}$ otherwise.

Note that we only specify the parts of the algebra we actually need for our example, all others can be picked arbitrarily. If we want to evaluate $\boxed{\forall x.x = x}$ in A, recall that $\overline{\sigma} = \varnothing$ and thus $\mathcal{V} = \Lambda p \in \overline{\sigma}.[\![\varnothing(p)]\!]^A_\varnothing = \varnothing$, so we have

$$[\![\mathbf{U}]\!]^A_\varnothing = [\![\mathbb{B}(\mathbb{S}(\forall)\,[x].\boxed{x = x})]\!]^A_\varnothing = \beta^A_\varnothing(\mathsf{q}, \varnothing, \mathcal{F})$$

where $\mathcal{F} = \Lambda u \in U^A.[\![\boxed{x = x}]\!]^A_{[x/u]}$. So $[\![\mathbf{U}]\!]^A_\varnothing = \mathsf{t}$, iff $\mathcal{F}(u) = \mathsf{t}$ for all $u \in \mathbb{N}$. But observe that we have $\mathcal{F}(u) = [\![\mathbb{A}(=, \mathbb{V}(x), \mathbb{V}(x))]\!]^A_{[x/u]} = @^A_2(\mathsf{e}, \langle u, u\rangle) = \mathsf{t}$ by definition, and thus $[\![\mathbf{U}]\!]^A_\varnothing = \mathsf{t}$ as expected.

Extending A to an interpretation of the λ-binder is more complicated because we have to commit to a type theory.

Example 4 (Continuing Example 2). We extend U^A so that it contains all function sets that can be formed from the natural numbers, i.e., $\mathbb{N}^{\mathbb{N}}$ $\mathbb{N}^{(\mathbb{N}^{\mathbb{N}})}$, $(\mathbb{N}^{\mathbb{N}})^{\mathbb{N}}$ and so on, as well as the functions they contain we call this set $\mathbb{N}^{**}$. For this to be useful, we should also extend our vocabulary with symbols ι and $\rightarrow$. We put

1. $U := \mathbb{N}^{**} \cup \{\mathsf{l}, \mathsf{p}\}$
2. $R^A_n = U^{(U^n)}$,
3. $\iota^A = \mathbb{N}$, and $\rightarrow^A = \mathsf{p}$, and
4. interpret $@^A_2(\mathsf{p}, \langle u, v\rangle)$ as the set of functions from v to u if u and v are sets and as f otherwise. Furthermore, we put $@^A_2(f, u) = f(u)$ whenever function application is defined.
5. Then for $\overline{\sigma} = \{\langle 1, 1\rangle\}$, $K = \{\langle 1, 1\rangle \mapsto \tau\}$, we can put $\beta^A_K(\mathsf{l}, \mathcal{V}, \mathcal{F})$ to be the function $\Lambda u \in \mathcal{V}(1, 1).\mathcal{F}(u)$.
6. $\alpha^A_\tau(u, v) = u$.

Then we can interpret $\boxed{\lambda x : \beta.x}$ as follows. We have $[\![\mathbf{L}]\!]^A_\varnothing = [\![\mathbb{B}(\mathbb{S}(\lambda)\,[x|K := V].\mathbb{V}(x))]\!]^A_\varnothing = \beta^A_\varnothing(\mathsf{l}, \langle \mathcal{V}, \mathcal{F}\rangle)$ where

- $\mathcal{V} = \Lambda p \in \{\langle 1, 1\rangle\}.[\![V(p)]\!]^A_\varnothing = \Lambda p \in \{\langle 1, 1\rangle\}.[\![\boxed{\beta}]\!]^A_\varnothing$,
- $\mathcal{F} = \Lambda u \in U.[\![\mathbb{V}(x)]\!]^A_{[x/u]} = \Lambda u \in U.u$

And thus, we evaluate $\beta^A_\varnothing(\mathsf{l}, \mathcal{V}, \mathcal{F})$ as the identity function on $[\![\boxed{\beta}]\!]^A_\varnothing$ as expected.

A simple induction over the construction of OpenMath objects in Definition 4 using the respective clauses in Definition 10 gives us an OpenMath version of the well-known

Lemma 1 (Substitution Value Lemma). $[\![[x/o']o]\!]^{\mathcal{A}}_{\varphi} = [\![o]\!]^{\varphi}_{a,[x/[\![o']\!]^{\mathcal{A}}_{\varphi}]}$

This in turn can be specialized in the usual way to obtain:

Corollary 1 (Soundness of α-Equality). *If $o \equiv_\alpha o'$ then $[\![o]\!]^{\mathcal{A}}_{\varphi} = [\![o']\!]^{\mathcal{A}}_{\varphi}$.*

So we have shown that OM algebras form a model class for OpenMath objects. We will now show that they characterize them up to isomorphism. For that we need to consider initial models, which will function as canonical representatives in this model class.

Definition 11 (Free OM Algebra). Let T be an OM vocabulary. Let $\overline{Subs(o)}$ abbreviate the function $\Lambda\langle[o_1]_\alpha,\ldots,[o_n]_\alpha\rangle \in U^n.[Subs(o)\langle o_1,\ldots,o_n\rangle]_\alpha$. Then the **free OM algebra** $I := I(T)$ **over** T is defined as follows.

1. $U^I = O(T,\emptyset)/_{\equiv_\alpha}$, i.e. the quotient set of the ground OpenMath objects modulo α-conversion.
2. R^I_n is the set of functions $\overline{Subs(o)}$ for $o \in O(T,\langle x_1,\ldots,x_n\rangle)$,
3. $s^I = [\mathbb{S}(s)]_\alpha$,
4. $@^I_n([f]_\alpha,\langle[o_1]_\alpha,\ldots,[o_n]_\alpha\rangle) = [\mathbb{A}(f,o_1,\ldots,o_n)]_\alpha$,
5. for a binding signature σ: $\beta^I_K([b]_\alpha,\mathcal{V},\mathcal{F}) = [\mathbb{B}(b\,[x_1,\ldots,x_n \mid K := V].o)]_\alpha$ where
   - $V = \Lambda p \in \overline{\sigma}.v_p$ for some $v_p \in \mathcal{V}(p)$,
   - $o \in O(T,\langle x_1,\ldots,x_n\rangle)$ is some object such that $\overline{Subs(o)} = \mathcal{F}$.
6. $\alpha^I_k([o]_\alpha,[v]_\alpha) = [\mathbb{K}(o|k:=v)]_\alpha$.

Lemma 2. *$I(T)$ is well-defined.*

Proof. We need to show several well-definedness conditions.

$\overline{Subs(o)}$: Substituting ground α-equivalent objects preserves α-equivalence. This follows from the definition of α-equivalence.

$@^I_n$: If $f \equiv_\alpha f'$, $o_1 \equiv_\alpha o'_1,\ldots,o_n \equiv_\alpha o'_n$, then $\mathbb{A}(f,o_1,\ldots,o_n) \equiv_\alpha \mathbb{A}(f',o'_1,\ldots,o'_n)$. This follows directly from the definition of α-equivalence.

β^I_K: If $b \equiv_\alpha b'$, $v(p) \equiv_\alpha v'(p)$ for all $p \in \overline{\sigma}$, then there exists an $o \in O(T,\langle x_1,\ldots,x_n\rangle)$ such that $\overline{Subs(o)} = \mathcal{F}$, and for two such o, o' we have that

$$\mathbb{B}(b\,[x_1,\ldots,x_n|K := \Lambda p.v_p].o) \equiv_\alpha \mathbb{B}(b'\,[x_1,\ldots,x_n|K := \Lambda p.v'_p].o').$$

The existence follows from the definition of R^I_n. The α-equivalence holds because non-α-equivalent objects induce non-α-equivalent substitution functions.

α^I_k: If $o \equiv_\alpha o'$ and $v \equiv_\alpha v'$, then $\mathbb{K}(o\,|\,k := v) \equiv_\alpha \mathbb{K}(o'\,|\,k := v')$. This follows directly from the definition of α-equivalence.

Lemma 3. *Let T be an OM algebra. Then $[\![o]\!]^{I(T)} = [o]_\alpha$ for every $o \in O(T,\langle\rangle)$.*

Proof. This is proved by a straightforward induction on the structure of o.

Lemma 4 ($I(T)$ is initial). *Let A be an OM algebra over T, and let $I := I(T)$. Then there is a (unique) mapping $h : U^I \to U^A$ satisfying $h([\![o]\!]^{I(T)}) = [\![o]\!]^{\varphi}$.*

Proof. This follows directly from Cor. 1 and Lem. 3.

Corollary 2 (Completeness of α-Equality). *If $[\![o]\!]^A_\varphi = [\![o']\!]^A_\varphi$ for all OM algebras A, then $o \equiv_\alpha o'$.*

Proof. This follows from Lem. 3 by putting $A := I(T)$.

2.3 OpenMath Objects with Uninterpreted Symbols

The semantics discussed so far was based on the abstract notion of OM Vocabularies. To arrive at a semantics of OpenMath objects we need to relate this to OpenMath CDs.

The OpenMath2 standard introduces "abstract content dictionaries" to abstract from the concrete XML encoding of content dictionaries. According to [BCC+04, section 4.2], (abstract) CDs have a **CD name**, a **CD base URI**, and contain **symbol definitions**, which in turn consist (among others) of a **symbol name**, an optional **symbol role** (one of "binder", "attribution", "semantic-attribution", "application", "constant", and "error"), and a set of **mathematical properties**.

Definition 12 (OpenMath Symbols). We say that a CD C **declares** an OpenMath symbol $\langle n, c, u, r\rangle$, iff the CD base of C is u, the CD name of C is c, and C has a symbol definition with symbol name n and symbol role r (note that the role can be undefined as it is optional). We define the set *Symbols* to be the set of symbols declared by some OpenMath CD and the set *Keys* to be those with symbol role "semantic-attribution".

There are three differences between abstract OM Objects and standard OpenMath objects; all three are related to symbols and keys:

1. We do not take keys to be abstract OM objects by themselves (see clause 1 in Definition 4). We claim that that there are no mathematically meaningful situations where keys can appear except in attributions. This design decision should not be perceived as a serious impediment for our semantics, since keys can be added analogously to the treatment below at the cost of adding an additional case everywhere.
2. The OpenMath2 [BCC+04] "role system", poses some additional restrictions on where symbols can occur, but not enough to simplify our construction of binding signatures. Therefore, we disregard it here and refer the reader to [RK09a] for details and an extended role system proposal that would.
3. We do not consider attributions with symbols that are not in *Keys*, in particular symbols with roles "attribution" which are intended by the OpenMath2 standard for just this purpose. However the standard states

> *This form of attribution may be ignored by an application, so should be used for information which does not change the meaning of the attributed OpenMath object.* [BCC+04, clause 2.1.4.*ii*]

and therefore it necessary to disregard these attributions in the construction of a semantics for OpenMath. In the mapping from standard OpenMath objects to abstract ones, we strip attributions with non-*Keys* symbols.

This allows us to define the meaning of an OpenMath object. As we are not taking mathematical properties in CDs into account, we will think of these symbols as uninterpreted, therefore we will call it the "algebraic meaning".

Definition 13 (Algebraic Meaning). Let o be an OpenMath object, then we call the set of symbols such that $\mathbb{S}(s)$ occurs in o the **OM vocabulary induced by** o.

If $o \in O(T, \langle\rangle)$ is a ground OM Object, T its induced vocabulary, and A an OM algebra over T, then the **algebraic meaning of** o **in** A is $[\![o]\!]^A$ and the **algebraic meaning of** o is $[\![o]\!]^{I(T)}$.

Note that the algebraic meaning of an abstract OpenMath object is just its (standard) OpenMath object.

As discussed in the introduction, the algebraic semantics only gives us a rather weak and syntactic concept of meaning of the OpenMath language. To understand the full meaning of OpenMath objects we need to take CDs into account, which we do in the next section.

3 OpenMath Models

If we want to understand mathematical properties in OpenMath content dictionaries, we need to have a notion of "truth" — after all the properties are assumed to hold. Furthermore, we need to take into account the mathematical properties themselves. In OpenMath there are two kinds of mathematical properties: "commented mathematical properties" (encoded as `CMP` elements which contain mathematical vernacular) and "formal mathematical properties" (encoded as `FMP` elements that contain XML encodings of OpenMath objects). We are going to concentrate on the latter in this paper since they provide more structure. This is no loss of generality, given the assumption in mathematical practice that any rigorously stated property can be fully formalized given enough resources. For for the purposes of this paper we will just assume that we have access to an oracle that translates all commented mathematical properties into formal ones, which we handle with the methods presented in this section.

3.1 Theories and Satisfaction

As formal mathematical properties are expressed as OpenMath objects, we will need to build the required notion of "truth" into an OM vocabulary, which is rather simple.

Definition 14 (OM Logic). An OM vocabulary L with distinguished symbols $\top$ and $=$ is called an **OM logic**.

In OpenMath CDs, (formal) mathematical properties are expressed as statements in some foundational logical system, thus the OM Objects representing them will in general contain symbols from the foundation and the CD itself. For instance, the `arith1` CD [CDa04] contains an `FMP` with the object $\boxed{\forall a, b.a + b = b + a}$ to express commutativity of addition. The symbols $\boxed{\forall}$ and $\boxed{=}$ are from the vocabulary of the foundational system and the symbol $\boxed{+}$ is from the CD itself.

We will treat OpenMath content dictionaries as logical theories, which are determined by their vocabularies and axioms, and model them using institutions (see [Rab08] for an introduction to both).

Definition 15 (Theory). Let L be an OM logic and T an OM vocabulary. An **OM theory** Θ for L is a pair $\langle T, Axioms(\Theta)\rangle$ where $Axioms(\Theta) \subseteq O(L+T, \langle\rangle)$. We will use $O(\Theta, C) := O(L + T, C)$ and take an OM algebra over Θ to be an OM algebra over $L + T$.

Note that $\langle \varnothing, \varnothing\rangle$ is a theory for any OM logic L, we call $\langle \varnothing, \varnothing\rangle$ the **empty theory over** L.

In this setting we can define OM models as those algebras that respect equality and in which the axioms hold.

Definition 16 (Model). Let L be an OM logic and Θ be an OM theory for L. An OM algebra M over Θ is a **model** of Θ if

- for all V, $o, o' \in O(T, V)$ and φ, we have that $[\![\mathbb{A}(=, o, o')]\!]^M_\varphi = [\![\top]\!]^M$ iff $[\![o]\!]^M_\varphi = [\![o']\!]^M_\varphi$,
- for all $A \in Axioms(\Theta)$, we have that $[\![A]\!]^M = [\![\top]\!]^M$.

The **Model Class** $\mathcal{M}(C)$ **of** Θ is the set of OM Models of Θ.

This gives us the standard notions of satisfaction and semantic entailment.

Definition 17 (Satisfaction). Let L be an OM logic, Θ be an OM theory for L, $o \in O(\Theta, V)$, M an OM model of Θ, and φ an assignment for V into M. Then we say that M **satisfies** o **under** φ (which we denote as $M, \varphi \models o$), iff $[\![o]\!]^M_\varphi = [\![\top]\!]^M_\varphi$. We write $M \models o$ if $M, \varphi \models o$ holds for all assignments φ and say that o is *valid* in M.

Definition 18 (Entailment). Let Θ be an OM theory and o a ground object. Then we say that Θ **entails** o ($\Theta \models o$), iff $M \models o$ for all $M \in \mathcal{M}(\Theta)$.

Example 5 (Continuing Example 3). We can make the vocabulary $\{\forall, =\}$ into an OM logic by adding $\top$ as the distinguished symbol, and the OM algebra A from 3 into a model (for the empty theory over $Q := \{\forall, =, \top\}$) by setting $\top^A := \mathtt{t}$. Note that $\mathbf{U}$ is entailed by the empty theory over Q.

3.2 The Meaning of OpenMath CDs and Objects

Note that the definitions above are still abstract in the sense that they refer to OM vocabularies, and OM theories, and not OpenMath CDs. So as in section 2.3 we have to relate abstract OM objects to standard ones and in particular to answer the question: what is the theory of a content dictionary? The OpenMath2 standard leaves this information under-defined, so we propose an interpretation that allows us to define an adequate notion of mathematical semantics[5].

Note that OpenMath CDs need not be self-contained, i.e. their `FMP`s can contain symbols that are neither introduced in the CD nor from the foundational system. Of course, these symbols (and thus the CDs that introduce them) should have an effect on the meaning of the symbols described by the `FMP`, so they need to be taken into account; naturally this process must be iterated until fixed point has been reached.

Definition 19 (CD Import). Let C be an OpenMath content dictionary, then we say that C **imports** D, iff $C \neq D$ and some `FMP` element in C contains a symbol with CD D. We call a CD **basic**, iff it does not import other CDs.

In contrast to other module systems for Mathematics (see [RK08,RK09b] for an overview) OpenMath does not make make the "imports relation" explicit and in particular does not make any assumptions about the absence of cycles.

Definition 20 (Signature and Property set of a CD). The **signature** of a CD C is the set of symbols it declares in union with the signatures of all CDs imported by C.

Similarly, the **property set** of a CD C is the set of OpenMath objects in `FMP` elements in C (these are called the **local properties** of C) in union with all the axiom sets of all CDs imported by C.

With this, we can directly define the OM theory induced by a CD.

Definition 21 (Theory of a CD). We call the pair $\langle S, P\rangle$, where S is the signature of C and P is the property set of C the **OM theory of** C.

In essence, the OM theory of a content dictionary is the union of all symbol declarations and mathematical properties from all theories from which a symbol is used in the CD. There is no problem with the (implicit) imports being cyclic, since their morphisms are the identity and we are constructing the (iterated) union. Note furthermore that OpenMath only supports literal CD names, and we can assume the set of CDs to be finite, therefore, the signature and axiom set of a CD are finite.

Note that in contrast to our definitions from section 3.1, the signature of a CD will already contain the OM logic, as OpenMath does not distinguish OM

[5] Arguably the OpenMath standard cannot fix this fully, since it intends to support all mathematical software systems including such that are "semantics-independent" like mathematical editing systems.

logics from other CDs. Following accepted mathematical practice we assume the logic to be first-order logic (with a choice operator) and a version of axiomatic set theory as a theory of first-order logic — we choose Zermelo Fraenkel set theory with choice [Zer08,Fra22] since this is the best-known one. Note that any other foundation of Mathematics would serve equally well for our purposes. For simplicity of presentation we will assume the existence of two basic CDs for first-order logic (declaring connectives, quantifiers, equalities, and choice) and `ZFC` (declaring membership and axioms).

In OpenMath practice, commented mathematical properties seem to assume ZFC as a foundational system, whereas `FMP`s make due with less: they usually only use symbols from the CDS

- `logic1` [CDl04]: a logic in the sense of Definition 14, as it supplies the symbol `true` — which we take as the distinguished symbol $\top$,
- and the symbol `eq` from CD `relation1` [CDr04] — which we take as $=$,
- `quant1` [CDq04] that supplies the first-order quantifiers.

Definition 21 allows us to define the meaning of a CD as a class of OM models.

Definition 22 (Model Class and Entailment for CDs). Let C be an OpenMath CD and Θ the OM theory of C, then the **Model Class of** C is $\mathcal{M}(\Theta)$ and $C \models o$, iff $\Theta \models o$.

We will now turn to the initial semantics again, this time to build initial OM models.

Definition 23 (Congruence Relation). Let T be a OM vocabulary and A an OM algebra over T. A **congruence relation on** A is a family of equivalence relations on U^A and R^A_n all denoted by $\equiv$ such that (whenever applicable)

1. if $u \equiv u'$ and $u_i \equiv u'_i$ for $i = 1, \ldots, n$, then
$$@^A_n(u, \langle u_1, \ldots, u_n\rangle) \equiv @^A_n(u', \langle u'_1, \ldots, u'_n\rangle),$$
2. for $l(\sigma) = n$, if $u \equiv u'$, $\mathcal{V}(p) \equiv \mathcal{V}'(p)$ for all $p \in \overline{\sigma}$, and $\mathcal{F} \equiv \mathcal{F}'$, then
$$\beta^A_K(u, \mathcal{V}, \mathcal{F}) \equiv \beta^A_K(u', \mathcal{V}', \mathcal{F}'),$$
3. if $u \equiv u'$ and $v \equiv v'$, then $\alpha_k(u, v) \equiv \alpha_k(u', v')$,
4. if $\mathcal{F} \equiv \mathcal{F}'$ and $u_i \equiv u'_i$ for $i = 1, \ldots, n$, then
$$\mathcal{F}(\langle u_1, \ldots, u_n\rangle) \equiv \mathcal{F}'(\langle u'_1, \ldots, u'_n\rangle).$$

Definition 24 (Quotient Algebra). Let T be an OM vocabulary, A an OM algebra over T, and $\equiv$ a congruence relation on A. Then the OM algebra $Q := A/\equiv$ over T is defined by:

1. $U^Q = U^A/\equiv$,

2. R_n^Q is the set of all functions of the form

$$f : (U^Q)^n \to U^Q,\ f([u_1]_\equiv, \ldots, [u_n]_\equiv) = [F(u_1, \ldots, u_n)]_\equiv$$

   for some $F \in R_n^A$,
3. $@_n^Q$, β_K^Q, and α_k^Q are induced by their analogues in A.

Lemma 5. *In the situation of Def. 24,*

- *Q is a well-defined OM algebra if A is,*
- *for all Q-assignments φ and A-assignments φ' such that $[\varphi'(x)]_\equiv = \varphi(x)$ for all variables x, it holds that $[\![o]\!]^Q_\varphi = \left[[\![o]\!]^A_{\varphi'}\right]_\equiv$.*

Proof. To prove the first part of the lemma, we have to assume a context C, an assignment φ, $o \in O(T, C)$, and $n \in \mathbb{N}$, and then show that $\Lambda\langle v_1, \ldots, v_n\rangle \in (U^Q)^n.[\![o]\!]^Q_{\varphi,[x_1/v_1,\ldots,x_n/v_n]} \in R_n^Q$. Let this be (1), and let (2) be the second part of the lemma. Then we can prove (1) and (2) in a joint induction on o.

The induction step for (1) follows if we show that there is an $F \in R_n^A$ such that for all $u_1, \ldots, u_n \in U^A$ it holds that $[\![o]\!]^Q_{\varphi,[x_1/[u_1]_\equiv,\ldots,x_n/[u_n]_\equiv]} = [F(u_1, \ldots, u_n)]_\equiv$. And using the induction hypothesis for (2), this follows from the well-definedness of A.

For all cases except variables and symbols, the induction step for (2) follows immediately from the definition of congruence. For variables, it follows immediately from the relation between φ and φ'. For symbols, it is trivial.

Definition 25 (Induced Congruence). Let $\Theta = \langle T, Ax\rangle$ be an L-theory, then we define a congruence relation $\equiv_\Theta$ on $I(L + T)$ as follows:

$$[o]_\alpha \equiv_\Theta [o']_\alpha \quad \text{iff} \quad \Theta \models \mathbb{A}(=, o, o') \quad \text{for } o, o' \in O(L + T, \langle\rangle)$$

and

$$\overline{Subs(o)} \equiv_\Theta \overline{Subs(o')} \quad \text{iff} \quad \Theta \models \mathbb{A}(=, o, o') \quad \text{for } o, o' \in O(L{+}T, \langle x_1, \ldots, x_n\rangle).$$

We call $\equiv_\Theta$ the **congruence induced by** Θ.

Lemma 6. *Let $\Theta = \langle T, Ax\rangle$ be an L-theory, then $\equiv_\Theta$ is indeed a congruence relation.*

Proof. The proof is straightforward.

As a consequence, the following construction is well-defined.

Definition 26 (Initial Model). Let $\Theta = \langle T, Ax\rangle$ be an L-theory, then $I(\Theta) := I(L + T)/\equiv_\Theta$ is called the **initial model for** Θ.

And that finally yields

Theorem 1. *For all $o \in O(\Theta, \langle\rangle)$ we have $I(\Theta) \models o$ iff $\Theta \models o$. In particular, $I(\Theta)$ is a Θ-model.*

Proof. We know $I(\Theta) \models o$ iff $[\![o]\!]^{I(\Theta)} = [\![\top]\!]^{I(\Theta)}$. Using Lem. 5, this is equivalent to $[[\![o]\!]^{I(T)}]_{\equiv_\Theta} = [[\![\top]\!]^{I(T)}]_{\equiv_\Theta}$ where T is the vocabulary of Θ. The latter is equivalent to $\Theta \models \mathbb{A}(=, o, \top)$ by Lem. 3 and Def. 25. And that is equivalent to $\Theta \models o$.

And that yields the main theorem of this section

Corollary 3 (Herbrand Theorem). *Every OM theory has a model that arises as a quotient of the free OM algebra.*

Note that this is exactly the bridging result between the OpenMath objects semantics postulated in the OpenMath2 standard (see Section 1.3) and the traditional foundations of Mathematics (see section 1.2). And with that we can finally define the meaning of OpenMath objects.

Definition 27 (The Meaning of an OpenMath Object). Let o be an OpenMath object, then we call the union of the theories of the CDs referenced in o the **Theory** of o. If $o \in O(T, \langle\rangle)$ is a ground OM Object, Θ its theory, and M an OM Model of Θ, then the **meaning of o in M** is $[\![o]\!]^M$.

4 Conclusion

In this paper we have tried to rectify common misunderstandings about the meaning of OpenMath (and thus MathML3) expressions. A central point in the argument can be elucidated by another quote from the referee report mentioned in the introduction — it continued with

> *The [...] "free algebra" semantics is nonsense: it amounts to saying that "the meaning of a term is its syntax". That is not what a mathematical semantics is.* anonymous referee for [DK09]

We have shown that the free algebra of OpenMath objects forms an initial algebra for "formulae with uninterpreted symbols" which is syntactic in nature as all initial algebras are. Indeed for OpenMath and content MathML expressions that do not contain symbols — and are thus unrestricted by content dictionaries — this is the best meaning we can hope for: OpenMath cannot impose more restrictions than α-equivalence and flattening of attributions without losing coverage. Indeed this is captured by the the algebraic semantics of OpenMath expressions in Section 2.

But the meaning of an OpenMath object comes mainly from the mathematical properties in the content dictionaries of its symbols. In section 3 we have been able to show that this can be grafted onto the algebraic semantics by interpreting OpenMath CDs as logical theories over a foundational system like first-order logic with ZFC as an axiomatic set theory.

Note that our semantic analysis has only taken into account symbol names, roles, and mathematical properties. The former two are relevant for the OM vocabularies and the latter for the OM theories that give OpenMath symbols

their meaning. In particular, we did not look at descriptions (for symbols or whole CDs) or examples. The status of these CD parts is left unspecified, by the OpenMath2 standard, and usage in actual CDs is non-uniform. Symbol descriptions reach from appealing to the folklore — e.g. "*This symbol represents the Boolean value true.*" [CDl04] to specific literature references e.g. "*See CRC Standard Mathematical Tables and Formulae, editor: Dan Zwillinger, CRC Press Inc., 1996, (7.7.11) section 7.7.1.*" [CDs04]. Arguably both forms "mean" something to the human reader, and especially the latter should surely contribute to the theory. The case of examples in CDs is similarly unclear: if they were uninformative to the human reader, nobody would put them in. But again practice in published CDs is no help: examples are often statements — and thus in principle mathematical properties — about (mathematical objects constructed by) the symbols they illustrate, and — if they are — they tend to be valid, but it would be uncautious to assume this to be generally the case. The next version of the OpenMath standard could of course clarify these issues at the cost of making it more heavyweight and thus arguably less useful. We propose to use the OMDoc format [Koh06] that already makes these issues for specifying content dictionaries instead if the additional functionality is desired.

A final objection often brought up against the "semantics of OpenMath" is that the standard CDs maintained by the OpenMath society are very weak, and (even with the methods presented here) do not give a clear and unambiguous meaning for K-14 mathematics. Indeed this criticism is formally justified, but misses the main point of the OpenMath philosophy, namely that the set of CDs is open-ended, and that we can build CDs to suit all our communication and representation needs. In particular it is possible (and in fact rather simple) to build a CD `NatArith` for natural numbers and arithmetic by encoding the Peano Axioms and recursive equations for the arithmetical operators in OpenMath objects so that that its theory $\Theta = \Theta(\texttt{NatArith})$ determines the class of Θ-models up to isomorphism (and all are isomorphic to $\mathbb{N}$). To see this just use the standard proof with our notion of OM models from section 3. If this does not count as clear and unambiguous meaning then what? The OpenMath society (and the W3C Math Working Group for that matter) view the weakness of the standard OpenMath/MathML CD group as a feature and not a bug. These CDs contain fewer mathematical properties to allow them to describe larger model classes. For instance the CD `arith1` [CDa04] (somewhat) corresponds to the class of (Abelian) semigroups. And that is a good thing, since it is intended to capture the informal usage in K-14: in many situations we need the flexibility offered by the OpenMath/MathML CDs so that we do not over-specify the meaning. We would probably not want to scare elementary school children who are struggling with long division with the Peano Axioms or teenagers in high school with the fine differences between Riemann and Lebesque integration.

We end this treatise on the "meaning of OpenMath and MathML" with the observation that it is possible to specify the meaning of mathematical objects and formulae at many levels of flexibility and rigorousness and extend the invitation to our readers to do just that: to contribute content dictionaries

to the community of mathematicians (by way of the OpenMath society CD site [OMC08]).

References

ABC+03. Ron Ausbrooks, Stephen Buswell, David Carlisle, Stéphane Dalmas, Stan Devitt, Angel Diaz, Max Froumentin, Roger Hunter, Patrick Ion, Michael Kohlhase, Robert Miner, Nico Poppelier, Bruce Smith, Neil Soiffer, Robert Sutor, and Stephen Watt. Mathematical Markup Language (MathML) version 2.0 (second edition). W3C recommendation, World Wide Web Consortium, 2003.

ABC+09. Ron Ausbrooks, Bert Bos, Olga Caprotti, David Carlisle, Giorgi Chavchanidze, Ananth Coorg, Stéphane Dalmas, Stan Devitt, Sam Dooley, Margaret Hinchcliffe, Patrick Ion, Michael Kohlhase, Azzeddine Lazrek, Dennis Leas, Paul Libbrecht, Manolis Mavrikis, Bruce Miller, Robert Miner, Murray Sargent, Kyle Siegrist, Neil Soiffer, Stephen Watt, and Mohamed Zergaoui. Mathematical Markup Language (MathML) version 3.0. W3C Working Draft of 4 June 2009, World Wide Web Consortium, 2009.

AvLS98. John Abbott, Andre van Leeuwen, and Andreas Strotmann. Openmath: Communicating mathematical information between co-operating agents in a knowledge network. *Journal of Intelligent Systems*, 8, 1998.

Bar91. H. Barendregt. Introduction to Generalized Type Systems. *Journal of Functional Programming*, 1(2):125–154, 1991.

BBK04. Christoph Benzmüller, Chad Brown, and Michal Kohlhase. Higher order semantics and extensionality. *Journal of Symbolic Logic*, 69:1027–1088, 2004.

BCC+04. Stephen Buswell, Olga Caprotti, David P. Carlisle, Michael C. Dewar, Marc Gaetano, and Michael Kohlhase. The Open Math standard, version 2.0. Technical report, The Open Math Society, 2004.

Ber37. P. Bernays, 1937. Seven papers between 1937 and 1954 in the Journal of Symbolic Logic.

Bou64. N. Bourbaki. Univers. In *Sminaire de Gomtrie Algbrique du Bois Marie - Thorie des topos et cohomologie tale des schmas*, pages 185–217. Springer, 1964.

Can83. G. Cantor. Grundlagen einer allgemeinen Mannigfaltigkeitslehre. Ein mathematisch-philosophischer Versuch in der Lehre des Unendlichen. *Mathematische Annalen*, 1883.

CDa04. arith1. OpenMath Content Dictionary at `http://www.openmath.org/cd/arith1.ocd`, 2004.

CDl04. logic1. OpenMath Content Dictionary at `http://www.openmath.org/cd/logic1.ocd`, 2004.

CDq04. quant1. OpenMath Content Dictionary at `http://www.openmath.org/cd/quant1.ocd`, 2004.

CDr04. relation1. OpenMath Content Dictionary at `http://www.openmath.org/cd/relation1.ocd`, 2004.

CDs04. s_data1. OpenMath Content Dictionary at `http://www.openmath.org/cd/s_data1.ocd`, 2004.

CH88. T. Coquand and G. Huet. The Calculus of Constructions. *Information and Computation*, 76(2/3):95–120, 1988.

Chu40. A. Church. A Formulation of the Simple Theory of Types. *Journal of Symbolic Logic*, 5(1):56–68, 1940.

DK09. James H. Davenport and Michael Kohlhase. Unifying Math Ontologies: A tale of two standards. In Jacques Carette, Lucas Dixon, Claudio Sacerdoti Coen, and Stephen M. Watt, editors, *MKM/Calculemus 2009 Proceedings*, number 5625 in LNAI. Springer Verlag, 2009. in Press.

Fra22. A. Fraenkel. The notion of 'definite' and the independence of the axiom of choice. 1922.

Gir71. J. Girard. Une extension de l'interpretation de Gödel à l'analyse, et son application à l'élimination des coupures dans l'analyse et la théorie des types. In J. Fenstad, editor, *2nd Scandinavian Logic Symposium*, pages 63–92. North-Holland, 1971.

Göd31. K. Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme. *Monatshefte für Mathematik und Physik*, 38:173–198, 1931. English title: On Formally Undecidable Propositions Of Principia Mathematica And Related Systems.

Göd40. K. Gödel. The Consistency of Continuum Hypothesis. *Annals of Mathematics Studies*, 3:33–101, 1940.

HHP93. R. Harper, F. Honsell, and G. Plotkin. A framework for defining logics. *Journal of the Association for Computing Machinery*, 40(1):143–184, 1993.

Hil00. D. Hilbert. Mathematische Probleme. *Nachrichten von der Königlichen Gesellschaft der Wissenschaften zu Göttingen*, pages 253–297, 1900.

Hil26. D. Hilbert. Über das Unendliche. *Mathematische Annalen*, 95:161–90, 1926.

Koh06. Michael Kohlhase. OMDoc – *An open markup format for mathematical documents [Version 1.2]*. Number 4180 in LNAI. Springer Verlag, 2006.

ML74. P. Martin-Löf. An Intuitionistic Theory of Types: Predicative Part. In *Proceedings of the '73 Logic Colloquium*. North-Holland, 1974.

OMC08. OpenMath content dictionaries. web page at `http://www.openmath.org/cd/`, seen June2008.

Rab08. Florian Rabe. *Representing Logics and Logic Translations*. PhD thesis, Jacobs University Bremen, 2008.

Rey74. J. Reynolds. Towards a Theory of Type Structure. In *Paris Colloq. on Programming*, pages 408–425. Springer, 1974.

RK08. Florian Rabe and Michael Kohlhase. An exchange format for modular knowledge. In G. Sutcliffe, P. Rudnicki, R. Schmidt, B. Konev, and S. Schulz, editors, *Proceedings of the LPAR Workshops: Knowledge Exchange: Automated Provers and Proof Assistants, and The 7th International Workshop on the Implementation of Logics*, number 418 in CEUR Workshop Proceedings, pages 50–68, 2008.

RK09a. Florian Rabe and Michael Kohlhase. A better role system for openmath. submitted to the 22nd OpenMath Workshop, 2009.

RK09b. Florian Rabe and Michael Kohlhase. A web-scalable module system for mathematical theories. Manuscript, to be submitted to the Journal of Symbolic Computation, 2009.

Rus01. B. Russll. Recent work in the philosophy of mathematics. *International Monthly*, 1901.

Rus08. B. Russell. Mathematical Logic as Based on the Theory of Types. *American Journal of Mathematics*, 30:222–262, 1908.

Tar38. A. Tarski. Über Unerreichbare Kardinalzahlen. *Fundamenta Mathematicae*, 30:176–183, 1938.

vN25. J. von Neumann. Eine Axiomatisierung der Mengenlehre. *Journal für die reine und angewandte Mathematik*, 154:219–240, 1925.

WR13. A. Whitehead and B. Russell. *Principia Mathematica*. Cambridge University Press, 1913.

Zer08. E. Zermelo. Untersuchungen ber die Grundlagen der Mengenlehre I. *Mathematische Annalen*, 65:261–281, 1908. English title: Investigations in the foundations of set theory I.

A Better Role System for OpenMath

Florian Rabe, Michael Kohlhase

Computer Science, Jacobs University Bremen
{f.rabe,m.kohlhase}@jacobs-university.de

Abstract. OpenMath is a standard for the representation and communication of mathematical objects, which are built up from symbols and variables using applications, binding expressions, and key-value attributions. OpenMath2 introduced a set of symbol roles that can be specified in content dictionaries to restrict the occurrences of the respective symbols. This yields a simple, high-level notion of well-formed objects.
While this system is appealing in its simplicity, the definition of well-formedness is purely extensional without an intuitive or formal condition that distinguishes well-formed objects from ill-formed ones. Moreover, some well-formed objects should arguably rather be ill-formed. We try to remedy that with a refined role system while preserving the simplicity of the existing one. In particular, by distinguishing syntactic and semantic roles, we can capture the intuitive notion of well-formedness better.

1 Introduction

OpenMath is a standard for the representation and communication of mathematical objects, which are built up from symbols and variables as applications, binding expressions, attribution. To provide a simple, high-level well-formedness criterion the OpenMath2 standard [BCC+04] introduced a set of symbol roles that can be specified in content dictionaries to restrict the occurrences of the respective symbols. In the rest of this section we point out some problems of the OpenMath Role system before we will try to solve them in the Section 2. Section 4 concludes the paper.

1.1 General Problems of the OpenMath2 Role System

However, even though OpenMath attempts to use the role system for a high-level well-formedness check, its status remains somewhat unclear. The text says that

> *A symbol [...] cannot be used to construct a compound OpenMath object in a way which requires a different role (using the definition of construct given earlier in this section). This means that one cannot use a symbol which binds some variables to construct, say, an application object. [BCC+04, subsection. 2.1.4]*

the compliance chapter [BCC+04, Chapter 5] does not mention the role system at all. Furthermore, even the text above is not consequent enough to forbid clearly ill-formed expressions where binders and keys occur anywhere in an expression or where symbols without any role are used as binders or keys. Even worse: The standard permits composed objects as the first child of a binder; thus, any symbol can be used as a binder after all by wrapping it in a meaningless attribution.

As an example, for a well-formed OpenMath 2 expression that would be ill-formed under our proposal, consider the following where C is any symbol with the role `attribution`.

```
<OMBIND>
  <OMATTR>
    <OMATP>
      C
      <OMS cd="quant1" name="forall"/>
    </OMATP>
    <OMS cd="arith1" name="plus"/>
  </OMATTR>
  <OMBVAR><OMV name="x"/></OMBVAR>
  <OMS cd="sts" name="type"/>
</OMBIND>
```

It uses the `plus` symbol as a binder by wrapping it in an attribution that applications may ignore, uses a binder as the value of an attribution, and a key as the scope of a binder. Standard-compliant implementations or theoretic investigations must handle this object like any other one.

Thirdly, while it is reasonable to avoid a type system in OpenMath, it would be easily possible and extremely helpful to restrict the number of arguments that a symbol can be applied to.

1.2 Complex Binding Operators and Attribution Keys

In a nutshell, the OpenMath role system uses the roles `binder`, `error`, `attribution` and `semantic-attribution`, `application`, and `constant`. The underlying design principle of the OpenMath role system is that binders, errors, keys, and applications occur as the first children of `OMBIND`, `OME`, `OMATP`, and `OMA` objects. While this is certainly appealing, there are several disadvantages.

However, composed expressions must be allowed as applications in order to permit anonymous functions. Therefore, we have the choice to either permit composed binders, errors, and keys or to break the symmetry between the constructors.

Considering the former option, it is indeed often convenient to use binders and keys that are composed expressions, namely the results of applications. For instance we have found that integration can be expressed most elegantly if the integral is an operator that takes the domain of integration as the argument and returns the binder . For example,

```
<OMBIND>
  <OMA>
    <OMS cd="calculs1" name="integral"/>
    <OMA>
      <OMS cd="interval1" name="interval"/>
```

```
      <OMV name="a"/>
      <OMV name="b"/>
    </OMA>
  </OMA>
  <OMBVAR><OMV name="x"/></OMBVAR>
  f(x)
</OMBIND>
```

is a most natural representation of $\int_a^b f(x)dx$. Here and in the future, we will use boxed mathematical formulae to abbreviate OpenMath Objects wher the XML representations is immaterial to the exposition.

Keys are typically atomic but not always. For example, the typing relation in a language with an infinite type hierarchy is parametrized by an integer value for the type level.

Furthermore, both for binder and for keys, it is conceivable to use symbols wrapped in non-semantic attributions, i.e., to attach presentation information to them.

2 A two-dimensional Role System

Roles are associated with symbols in content dictionaries. We propose to generalize the role of a symbol into two orthogonal aspects:

- `role` represents the syntactic role of a symbol. It corresponds roughly to the `role` of OpenMath2.
- `arguments` represents the number of arguments that a symbol takes.

In this section, we will present the extended role system on a conceptual level and deal with syntactic issues in Section 3.

The `role` has three possible values:

- `term`: This role represents all kinds of expressions as they occur in mathematics and type theories. It is the default if no role is given.
- `binder`: This role represents binding operators. There is an informal consensus among mathematicians and computer scientists that expressions and binders are to be distinguished. The characteristic feature of binders is that they need variables and scope and cannot occur alone. For example, in lambda calculus almost everything is an expression but not the λ itself (and not the application operator, which is present in OpenMath already anyway). Similarly, every mathematician would interpret a $\int$ symbol occurring by itself as the non-binding operator on functions and never as a binder.
- `attribution` and `semantic-attribution`: These roles represent keys that can occur in attributions. Just like binders, they do not carry mathematical meaning on their own and only become meaningful within an attribution.

We will abbreviate these four values as $\mathcal{T}$, $\mathcal{B}$, $\mathcal{A}$, and $\mathcal{S}$, respectively.

There is no value `error` because we hold that the property of being an error is not a syntactic property like those of being a binder or a key. Rather, it is a

semantic property. This is confirmed by programming languages such as Java or SML where exceptions are treated as normal expressions that only obtain their special semantics in the type-checking and execution phase. Therefore, we argue that `OME` objects should be abandoned in favor of `OMA` objects. The property of being an error should be marked up by introducing a second role attribute for semantic roles. This new attribute is not only useful to mark up errors, but can also be used to mark up other semantic roles such as element, sort, proof, or judgment. We come back to this in Sect. 2.2.

There is no value `application` either. This is because the argument why binders and keys should be separated from expressions does not carry over to applications: A symbol designated as an application may very well occur separately and has a well-defined meaning if it does. For example, in the context of natural numbers, + has the set-theoretical meaning $\{((x, y), z) \mid x + y = z\}$.

We do not consider the property of constructing an application to be alternative to that of constructing a binder or a key. Rather do we consider it as an orthogonal property via the `arguments` attribute.

The attribute `arguments` has as values a natural number or the special value $*$. Its intended semantics is that it gives the number of arguments a symbol takes. In particular, by making the number of arguments 0 a symbol is forbidden from occurring as the first child of an `OMA` element. We also permit the special value $*$ to make the number of arguments unrestricted. If a symbol has the `semrole` of a binder or key, the default value of `arguments` is 0. This reflects the fact that binders and keys are typically atomic. If the `semrole` is `term`, the default value is $*$.

2.1 Well-formed Objects

We define the well-formed objects E and their syntactic roles $R(E) \in \{\mathcal{T}, \mathcal{B}, \mathcal{A}, \mathcal{S}\}$ in a mutual induction.

1. Every symbol $E = OMS(S)$ is a well-formed expression, and $R(E)$ is the value of the `semrole` attribute of S.
2. Every variable is a well-formed expression with role $\mathcal{T}$.
3. If $E, E_1, \ldots, E_n$ $(n > 0)$ are well-formed expressions, $R(E_i) = \mathcal{T}$ for all i, and either
   - E is a composed expression and $R(E) = \mathcal{T}$ or
   - E refers to a symbol and the value of that symbol's `arguments` attribute is $*$ or n,

   then $OMA(E, E_1, \ldots, E_n)$ is well-formed, and its role is $R(E)$.
4. If $E, V_1, \ldots, V_n, E'$ are well-formed expressions, $R(E) = \mathcal{B}$, $R(E') = \mathcal{T}$, and all V_i are well-formed attributed variables and $R(V_i) = \mathcal{T}$, then $OMBIND(E, (V_1, \ldots, V_n), E')$ is a well-formed expression with role $\mathcal{T}$.
5. If E, K, E' are well-formed expressions, $R(E) = \mathcal{T}$, $R(E') = \mathcal{T}$, and $R(K) \in \{\mathcal{S}, \mathcal{A}\}$, then $OMATTR(E, K := E')$ is a well-formed expression with role $R(E)$. (For simplicity, we omit the analogous case of multiple attributions.)

6. All elements of specific domains (numbers, strings) and all foreign objects are well-formed expression with role $\mathcal{T}$.

Note that we again omit the case of error objects. We come back to them in Sect. 2.2.

It is simple to make a RelaxNG schema out of the above definition. The schema can be generated from the CDs as in [Koh08].

Our role system satisfies the following invariants:

- Only terms may occur as arguments in applications, variables or scopes in bindings, or values in attributions.
- Keys and binders can only occur as the heads of attributions and bindings, respectively, and nothing else can occur in these positions.
- All symbols can take arguments, and the role of the symbol determines the role of the result. Symbols can be prevented from taking arguments and thus from occurring as the head of an application by making their number of arguments 0.
- All expressions can be attributed, and attributions do not change the role of the attributed expression.

In Case 5, it is reasonable to drop the requirement $R(E) = \mathcal{T}$, i.e., to permit attributions on binders and keys. However, that would make backwards compatibility somewhat harder when translating OpenMath 2 roles to our roles (see below).

2.2 Semantic Roles

In addition to distinguishing syntactic roles, it is often useful to give symbols with syntactic role $\mathcal{T}$ an additional semantic role attribute `semrole`. The intuition behind this becomes clear from the following list of possible values.

- `element` and `sort`: These roles represent mathematical objects and their containers.
- `proof` and `property`: These roles represent proof terms and properties of mathematical objects.
- `error` and `error-type`: These roles represent error objects and their containers.

In order to define the semantic role of an arbitrary term, we additionally permit the `semrole` attribute on an OMV element when occurring within an `OMBVAR` element. This is useful to give variables a semantic role. For example to distinguish between a lambda abstraction over elements or over types.

Both on variables and symbols, `element` is the default if no semantic role is given.

The semantic role of a well-formed expression with syntactic role $\mathcal{T}$ is defined as follows:

1. For a symbol: according to the declaration of the symbol.
2. For a variable: according to the declaration of the variable.
3. The semantic role of an application is that of the first child.
4. The semantic role of a binding is that of the third child.
5. The semantic role of an attribution is that of the attributed expression.
6. The semantic role of an element from a specific domain or foreign object is `element`.

Then the OME objects of OpenMath2 can be recovered as syntactic sugar for OMA objects with semantic role `error`.

2.3 Conservativity and Backwards Compatibility

The current roles of the OpenMath2 standard can be translated to ours as follows:

- `constant`: role $\mathcal{T}$ with 0 arguments,
- `application`: role $\mathcal{T}$ with $*$ arguments,
- `error`: role $\mathcal{T}$ with $*$ arguments and semantic role `error`.
- `binder`: role $\mathcal{B}$ with 0 arguments,
- `attribution`: role $\mathcal{A}$ with 0 arguments,
- `semantic-attribution`: role $\mathcal{S}$ with 0 arguments,
- no role: defaulting to role $\mathcal{T}$ with $*$ arguments.

Under this translation, we can prove the following restrictions for well-formed expressions in our sense over content dictionaries in the OpenMath2 sense:

- `constant`: This symbol may only occur by itself or with attributions: on toplevel, as an argument of an application, as the scope of a binding, or as a value of an attribution.
- `application` or `error`: This symbol may only occur as the first child of an application (possibly with attributions) or anywhere where symbols with role `constant` can occur.
- `binder`: This symbol may only occur as the first child of a binding.
- `attribution` or `semantic-attribution`: This symbol may only occur as the key of an attribution.
- no role: This symbol may only occur where terms may occur.

The last of this cases is the only backwards compatibility problem: In OpenMath 2, symbols with no role can occur anywhere, whereas we exclude such symbols from occurring as binders or keys. However, the number of OpenMath 2 symbols without a role that are meant to occur as a binder or key is so small (We do not know any example.) that we find that acceptable.

This means that OpenMath 2 content dictionaries can be translated to our role system in a way that well-formed expressions in our sense come very close to the apparently intended meaning of the OpenMath 2 standard.

3 Proposed Changes to the OpenMath Standard

We propose the following changes to the OpenMath standard.

1. Symbols declarations in content dictionaries have three attributes
   - `role`, values: `term` (default), `attribution`, `semantic-attribution`, `binder`,
   - `arguments`, values: natural numbers or *, * is default if `role` is `term`, 0 is default otherwise,
   - `semrole`, values: `element` (default), `sort`, `proof`, `judgment`, `error`, `error-type`.
2. OMV elements in variable declarations have a `semrole` attribute as above.
3. OMATP elements may have arbitrary objects as the first child.
4. The definitions of well-formed object, syntactic role of an object, and semantic role of a term from Sect. 2.1 are added to the standard and replace the existing descriptions of well-formed objects.
5. OME elements become syntactic sugar for OMA elements where the first child has semantic role `error`.

4 Conclusion

In this paper we have critically re-accessed the role system introduced in the OpenMath 2 standard. While this system is appealing in its simplicity, it has several drawbacks that we try to solve in this paper by generalizing roles into independent syntactic and semantic flavors.

This has the benefit that a straightforward and formal definition of well-formed objects can be achieved that preserves the generality and simplicity of OpenMath 2 while ruling out many so far permitted nonsensical objects. By adding the possibility of restricting the number of arguments of a symbol, users are able to succinctly restrict the possible uses of a symbol without incurring a significant gain in complexity.

Users can exploit our role system to characterize the possible first children of composed expressions more strictly as before, and these restrictions lead to invariants that are available to applications. Our role system would also tremendously simplify our definition of a set-theoretic semantics of OpenMath ([KR09]), which currently has to go out of its way to interpret practically useless objects.

Our extension is conservative in the sense that existing content dictionaries can be translated to our proposed system. Formerly well-formed objects stay well-formed except for those cases which the OpenMath2 role system – accidentally in our opinion – permitted.

References

BCC+04. Stephen Buswell, Olga Caprotti, David P. Carlisle, Michael C. Dewar, Marc Gaetano, and Michael Kohlhase. The Open Math standard, version 2.0. Technical report, The Open Math Society, 2004.

Koh08. Michael Kohlhase. Compiling OpenMath type systems to Relax NG grammars. In Olga Caprotti, Sebastian Xambó, Maria-Antonia Huertas, Michael Kohlhase, and Mika Seppälä, editors, *3rd JEM Workshop – Joining Educational Mathematics*, 2008.

KR09. M. Kohlhase and F. Rabe. Semantics of OpenMath and MathML3. In OpenMath *workshop*, 2009.

wiki.openmath.org – how it works, how you can participate

Christoph Lange

Computer Science, Jacobs University Bremen, ch.lange@jacobs-university.de

Abstract. At http://wiki.openmath.org, the OpenMath 2 and 3 Content Dictionaries are accessible via a semantic wiki interface, powered by the SWiM system. We shortly introduce the inner workings of the system, then describe how to use it, and conclude with first experiences gained from OpenMath society members working with the system and an outlook to further development plans.

1 Introduction: The OpenMath Content Dictionaries

OpenMath [17] is a semantic markup language ("content markup language") for mathematical formulæ that originated as a shared knowledge representation for applications in computer algebra and automated theorem proving in the mid-1990s and got further applied in areas as diverse as e-learning, scientific publishing, and interactive geometry. OpenMath defines an abstract data model for representing mathematical objects and two concrete syntaxes for it, a binary and a more common XML one. Important building blocks of mathematical objects are numbers, variables, symbols, and applications of mathematical objects to other mathematical objects. Any concrete operator, constant, set, or function can be a symbol. In contrast to, e. g., earlier versions of MathML, the symbol supply of OpenMath is constantly increasing due to its extensibility by so-called *content dictionaries* (CDs).

```
<CDDefinition>
<Name>plus</Name>
<Role>application</Role>
<Description>The symbol representing an n-ary commutative
 function plus.</Description>
<CMP>for all a,b | a + b = b + a </CMP>
<FMP>β(quant1#forall, a, b,
       @(relation1#eq, @(arith1#plus, a, b), @(arith1#plus, b, a)))</FMP>
</CDDefinition>
```

Fig. 1. Definition of the *arith1#plus* symbol

A CD is a collection of (usually closely related) mathematical symbols, each with a *name* and a mandatory informal *description* (cf. fig. 1). Further information

about symbols is optional but recommended to have: mathematical properties of the symbol, both in a formal (FMP) and an informal ("commented", CMP) flavour, and examples of applying the symbol. The language for expressing this information is part of the OpenMath standard. Besides the proper CD file (named e. g. `number-theory.ocd`), there can be additional files: OpenMath does not commit to a particular *type system*, so it allows for types of symbols to be specified in separate files parallel to the CD, one per type system. The most common type system in the OpenMath community is, however, Davenport's Small Type System (STS [3]); types in that system would be given in a file named `number-theory.sts`.

Furthermore, there is no doubt that *notations* must be specified for symbols in some way, if OpenMath objects should ever be presented to a human reader, but opinions diverge on whether this should be done in CD-like files or not. David Carlisle and others believe that directly writing XSLT (one file per CD, one template per symbol) does a good job in transforming OpenMath to Presentation MathML. The advantage of XSLT is its expressive power (it's Turing-complete!), which comes at the expense of human comprehensibility, though. Paul Libbrecht and Michael Kohlhase (of whose "camp" the author is a member) thus prefer CD-like dictionaries of XML-based notation definitions in a more compact syntax. They believe that, given a sufficient support for pattern matching or declarative symbol↦notation mappings, most, if not all aspects of mathematical notation can be handled, and authored much more intuitively. Libbrecht et al. *generate* XSLTs from notation definitions that use pattern matching, whereas Kohlhase et al. have implemented a dedicated renderer (actually two ones, which are being merged) that directly renders mathematical objects using either declarative or pattern-based notation definitions [8,7,9].

2 Authoring and Reviewing OpenMath CDs

While everybody is free do define his own CDs for his purposes, the OpenMath Society maintain a collection of *official* CDs [5] that have undergone a review process. Still, the content of an official CD is not fixed: It might still contain mistakes that have slipped through the review, or there might be ways to improve the informal descriptions of symbols, or relevant mathematical properties and examples to add.

As said in the introduction, one CD is essentially a file – containing several metadata fields on top, and then one *CDDefinition* block per symbol. The official CDs are maintained in a Subversion repository at `https://svn.openmath.org`. Developers participating in their maintenance check out a working copy of that repository, edit the CD files locally with a text or XML editor, and then commit their changes. RIACA have developed a Java-based CD editor [20], the only CD editor besides ours that we are aware of. The RIACA CD editor, however, rather focuses on generating Java code for programs dealing with OpenMath objects from CDs than on CD maintenance, and its development seems to have been discontinued for at least three years.

Issues with the CDs are usually being discussed on the OpenMath mailing list (om@openmath.org) in case of fixing bugs in existing CDs, or on the OpenMath 3 mailing list (om3@openmath.org) in case of the overhaul of the CDs and alignment with the Content MathML specification for the upcoming OpenMath 3 [4]. As an alternative for OpenMath 3, there is an installation of the Trac issue tracking system (cf. [24]) at https://trac.mathweb.org/OM3.

For presenting a CD to human readers, the elements of the OpenMath CD language are usually transformed to the desired output format (most commonly XHTML) using XSLT, and the OpenMath objects occurring inside the FMPs and examples are rendered as described in section 1. This presentation process is usually controlled by makefiles.

2.1 Three CD Editing Use Cases

In the remainder of this paper, I will focus on supporting three common use cases. First, the traditional way of handling these cases will be presented, to pave the way for showing how they are handled in the OpenMath wiki.

Minor Edits: Fixing minor mistakes does not change the semantics of a symbol. Consider correcting a spelling mistake in a description, or renaming a bound variable in a mathematical object that does not occur as a free variable in a subexpression. Supported by a text or XML editor only, which is not aware of the particular features of OpenMath CDs, such a fix would be done as follows (assuming that the mistake is in a CD from openmath.org):

1. Update the working copy of the OpenMath CDs
2. Open the CD file in question
3. Navigate to the *Description* child of the symbol in question
4. Fix the mistake
5. Commit the file (and, ideally: commit that file only, and give a meaningful log message that exactly refers to the symbol where the mistake was fixed)

Discussing and Implementing Revisions: Major revisions that change the semantics of a symbol have to be discussed among the developers before implementing them. Usually, the discussion starts with pointing out a problem (e. g. an FMP for a concrete symbol is wrong or misleading). Let us assume that the developer who identified the problem does not know how to solve it. Then, he would have to make others aware of the problem, e. g. by an e-mail to the OpenMath mailing list. Pasting a link to the Subversion URL of the CD in question into that e-mail helps others to inspect the problematic part[1]. Other developers would then reply to this e-mail and propose solutions, and again by replying to their mails, the solutions would be discussed, until the community agrees on one to be implemented.

[1] Trac features a more immediate and comprehensive integration of a trouble ticket system with a Subversion repository, but that is not currently possible for OpenMath, as the Trac and the Subversion repository are running on different servers.

Editing and Verifying Notations: Suppose that an example or FMP for a symbol σ in one CD uses a symbol τ from another CD and that the notation defined for τ is wrong. Concretely, imagine σ being the cumulative distribution function of the normal distribution, τ the integral symbol occurring in the definition of σ, and then imagine that the formatting of its lower and upper bounds is wrong. Here is how an author would fix this:

1. Identify the formal symbol name and CD of τ
2. Navigate to the file where the notation of τ is defined
3. Try to fix the notation definition
4. Regenerate the human-readable presentation of the CD defining τ (and, ideally: regenerate *all* CD presentations where τ occurs)
5. Open the regenerated presentation and check if it is correct (if not, back to 2)
6. Commit the file containing the notation definition, giving a meaningful log message

3 The OpenMath Wiki

From the previous use case descriptions it evident that a better tool support is needed to aid maintenance of the OpenMath CDs. SWiM is a wiki – a system for collaboration on knowledge collections on the web –, a *semantic wiki for mathematics* in particular [12]. It aims at offering intelligent collaboration services to authors of mathematical documents in semantic markup languages – such as OpenMath CDs. SWiM's notion of "semantics" is restricted to decidable structural aspects of documents and CDs; it does not capture the full semantics of OpenMath objects. Having presented first ideas at the OpenMath workshop in January 2008 [10], the author decided to further pursue supporting the OpenMath CD review as a case study for SWiM and set up an instance of the system at `http://wiki.openmath.org` in September 2008. Figure 2 shows a CD in the browsing view of SWiM. In the remainder of this section, it will be discussed how SWiM supports the use cases introduced in section 2.1.

3.1 Minor Edits

We have identified three different types of knowledge in OpenMath CDs: the structural outline of a CD (e. g. defining what symbols a CD defines), metadata (of such structural units, e. g. their informal descriptions or the date of revision), and OpenMath objects (inside FMPs and examples). For each of them, SWiM offers a dedicated editor (see [14]) for details.

It was a requirement for SWiM to allow for revisions in a context as local as possible – i. e. committing a "fixed description" to the CD repository instead of committing a "new revision of a CD with 'something' changed". SWiM acts as a browser and editor on top of the OpenMath Subversion repository but adopts a finer granularity. For a CD, there is not one lengthy wiki page, but, on every request of the CD from the Subversion repository, it is split into smaller

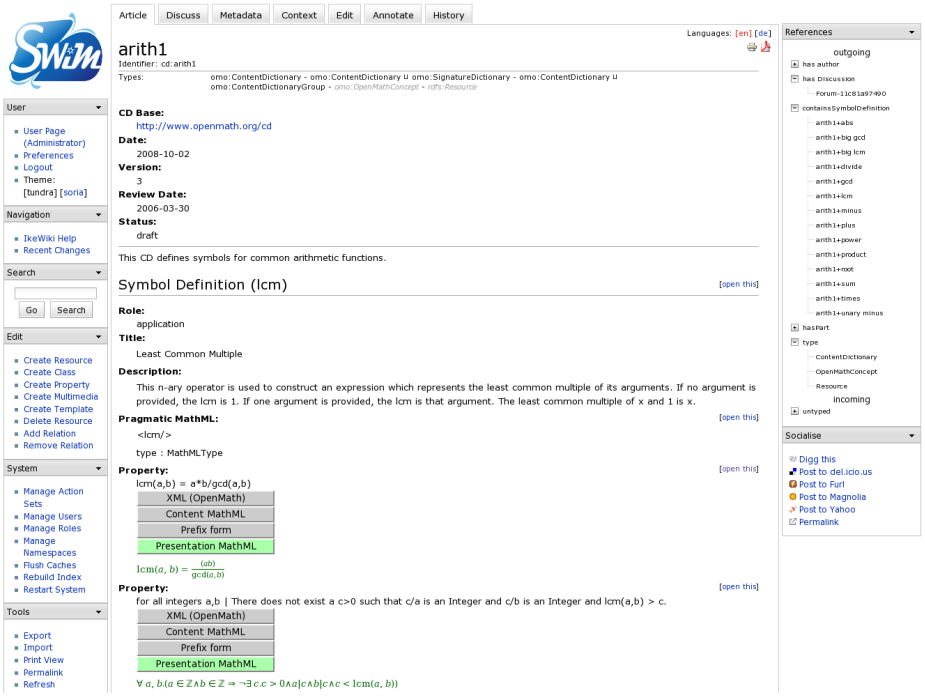

Fig. 2. An OpenMath CD in SWiM. Notice the navigation links on the right side.

logical units that are *semantically* subject to a revision: mathematical properties and examples on the lowest level, then symbol definitions (grouping several mathematical properties, examples, and metadata about one symbol together), and finally whole CDs. Of the wiki pages on CD and symbol definition levels, only the structural outline is editable, which keeps the content of the page editor small and maintainable; the smaller subparts that have been split into pages of their own right are editable separately and only represented as XInclude links [16] in the editing view. Nevertheless, a complete CD can be *viewed* at once; the presentation XSLTs have been adapted to cater for that. Metadata fields are either editable within the structural outline editor, or in a separate form-based view. Much attention was paid to avoiding any disruption of the file granularity of CDs in the Subversion repository, which are still editable in the conventional way[2]. Upon saving a change in the wiki, the whole CD to which the changed part belongs is reassembled, reversing the initial splitting process, and committed to the repository. However, the *log message* for this commit refers to the particular part of the CD that has been changed. In the revision log of the CD, such a revision will display as follows (here shown for a change of the description of the `transc1#sin` symbol):

```
r1234 | clange | 2009-05-11 13:06:41 +0200 (Mon, 11 May 2009) |
 2 lines
[Administrator@SWiM] replaced metadata field dc:description
Actually changed fragment cd:transc1+sin
```

The naming of CDs and parts thereof currently varies from OpenMath conventions and instead reflects the SWiM-internal RDF representation (as described in the following subsection) but could easily be adapted. The differing user names are owed to the technical limitation that SWiM and the Subversion repository do not have a unified user account management.

3.2 Discussing and Implementing Revisions

For each page (i. e. for each CD, symbol = CDDefinition, mathematical property, and example), SWiM offers a discussion page – essentially one local discussion forum per subject of interest. While that already allows discussions in the same granularity as our units of mathematical knowledge have, we have also given the discussion threads a semantic structure. On a conventional wiki discussion page, users would have to 1. manually create one section per discussion thread, 2. manually indent replies, 3. and point out the message of their discussion post in natural language. The IkeWiki platform [21] that SWiM is based on already cared for (1) and (2) by adopting the user interface known from discussion forums (and storing each discussion post as a separate resource instead of storing the whole discussion page, as other wikis do). We have added (3) in a way that

[2] As we will see in section 4, SWiM does have, and will always have, certain technical but also conceptual limitations, be they bugs or deliberate design choices, that disqualify it as a one-size-fits-all CD editor.

optionally allows users to indicate the *type* of their discussion posts in terms of an *argumentation ontology*, of which we present a simplified outline here (see [15] for details): Such a discussion can be started by pointing out a problem (here called *issue*). As replies to an *issue* post, *ideas* (= solution proposals) would be allowed, on which users can state their *position*, and finally a thread can be concluded with a post of type *decision*, summarising the idea that was actually agreed on to resolve the issue. For every possible type of reply to a discussion post, there is a dedicated reply button (cf. figure 3); "untyped" replies for posts that do not fit into this schema are still possible but obviously prevent further automated assistance.

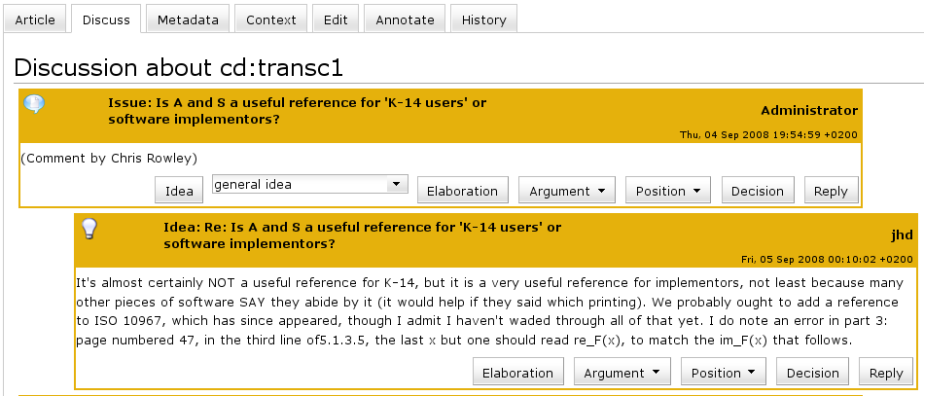

Fig. 3. Part of a discussion page from the OpenMath wiki. Notice the post types and the specialised reply buttons.

Aiming at a technical support that guides discussion threads towards common solutions, we added a domain-specific extension to the argumentation ontology. In a survey among OpenMath users[3], patterns of common problem and solution types in mathematical knowledge bases were identified [15]. The benefit from that is twofold: 1. Discussion threads can be queried by their logical structure, and 2. assistants for semi-automatically implementing common solution patterns to common problems can be implemented (cf. [15]). SWiM not only represents the structure of discussion threads in an RDF graph [19] in terms of the above-mentioned argumentation ontology, but it also represents the structure of CDs in terms of an ontology: part–whole links, as identified during the splitting of CDs described in section 3.1, links from symbol occurrences in mathematical objects to the place where they have been defined, as well as metadata. This whole RDF database can be queried. On the entry page of the OpenMath wiki, this is done

[3] The survey is still open for participation at `http://tinyurl.com/5qdetd` but likely to be replaced by a more focused survey soon.

in order to draw attention to unresolved issues by the following SPARQL [18] query:

```
SELECT DISTINCT ?P WHERE {
  ?P ikewiki:hasDiscussion ?D .
  ?C a arguonto:Issue;
     sioc:has_container ?D .
  OPTIONAL { ?Dec arguonto:decides ?C . }
  FILTER (!bound(?Dec)) }
```

P is a variable for a wiki page, which could be further restricted by its type in terms of the OpenMath ontology, e. g. we could restrict the query to symbols (*CDDefinition*). This query returns all pages P having a discussion forum D containing a comment `C` of type *Issue* on which no decision has been made so far. Such queries can be entered anywhere by an experienced user and result in a list of links to wiki pages.

3.3 Editing and Verifying Notations

In rendering mathematical objects to Presentation MathML, SWiM adopts the approach of the "Kohlhase camp" (cf. section 1) by embedding the JOMDoc rendering library [7,9] and maintaining notation dictionaries in parallel to content dictionaries. The notation definitions are browsable and editable in the wiki. The workflow of editing and verifying them, as outlined in section 2.1, is facilitated as follows (see [14,11] for details):

1. SWiM utilises the parallel markup [1, chapter 5.4] generated by the renderer to create links from the rendered symbols to the wiki pages representing their *CDDefinition*s. Thus, a developer can directly navigate from the occurrence of a symbol to its definition, and from there its notation definition is only one more click away.
2. The XHTML+MathML output of rendering a wiki page (= a CD or a fragment thereof) is cached, but after changing a notation definition of a symbol, the rendered output for all pages P containing a formula in which the symbol occurs is removed from the cache, forcing its re-generation. Note that the set P contains not only the FMP or example that immediately holds the OpenMath object using the symbol, but also the enclosing CDDefinition and CD. The set P is obtained by another SPARQL query on the database.

4 Discussion, Experiences and Further Directions

This section discusses the SWiM features presented so far, lists preliminary user feedback about them, as well as general feedback obtained from the users of the OpenMath wiki, and concludes with a schedule of plans for further improvement.

By supporting the use cases "minor edits", "discussing and implementing revisions" and "editing and verifying notations" and by its non-disruptive connection

to the OpenMath Subversion repository, SWiM facilitates crucial aspects of the CD maintenance process. Moreover, we got a fine-grained permission system for free from the underlying IkeWiki engine, which allows to define roles like "visitor" (may comment on everything), "CD editor" (may edit the CDs), and "administrator" (may also edit special pages like the entry page). The OpenMath developers have made little use of the wiki for actually *changing* the CDs (for usability reasons elaborated on below), but mainly used it as a *browser* – where is is slower but much richer in features than the statically rendered CD presentations –, and for *discussing*.

4.1 Evaluation

We have verified the principal utility of the basic argumentation ontology (without the domain-specific extensions yet) for OpenMath by importing an old corpus of e-mail conversations about the OpenMath/MathML 3 CDs by Chris Rowley, David Carlisle, Michael Kohlhase, and others, into the wiki, following the discussion structure. Further discussion posts have been contributed by OpenMath developers afterwards. Overall, this resulted in 90 discussion posts. A breakdown of this figure can be evaluated by post type and by post granularity:

by type: 69 posts fit into one of the types from the argumentation ontology, mainly *Issue* (48) and *Idea* (10). Only counting the 23 posts contributed by the users themselves (who were obviously less familiar with the background of the argumentation ontology), the result is slightly less convincing; for 9 of them the users were not sure how to classify them. The post type that was missing in most cases was nothing argumentative at all, but the *question* – either a direct question about some concept from a CD, or a follow-up question on an argumentative post, such as "what do you mean by this issue description?". It will be easy to solve that problem by adding such a post type. Some other posts could not be uniquely classified because they both raised an issue and proposed a solution (= idea) in the same sentence. Annotating different argumentative types not at the level of posts but *within* posts is highly non-trivial, both concerning conceptual modelling and user interface design, though, as discussed in [13].

by granularity: 36 posts (but only posts taken from the e-mail corpus) had individual symbols as their subject; the remaining 54 posts (including all of the posts made by users) were made on CD-level discussion pages. This shows that either the users did not find it intuitive (or not necessary) to access subparts of a CD when they saw a complete CD in the browser, or that it was not possible to identify individual symbols a post referred to. The latter is the case for certain posts that argue on design issues of a CD in general, sometimes naming certain individual symbols as examples. A few other posts from the e-mail corpus referred to *two* closely related symbols; we filed copies of them with both affected symbols.

Overall, this shows that the OpenMath CD editors have understood how to make use of this way of discussing problems, which is more exact than writing an e-mail or opening a Trac ticket.

The only evaluation of the editing features so far we have performed ourselves: We made sure that no content is lost or broken from the CD files in the Subversion repository during minor edits in SWiM. We have tested that by importing all OpenMath 3 CDs into the wiki, loading them into the editor once, saving them, and inspecting the XML diff.

A major criticism towards the wiki has so far been its focus on editing existing content. The different granularities of the wiki and the OpenMath Subversion repository make it very cumbersome to add, e. g., a new symbol to a CD: One has to edit the CD wiki page, add the new *CDDefinition* child there, as a sibling of the XInclude elements pointing to the existing *CDDefinition*s, and then save the CD page. Upon saving, the new *CDDefinition* fragment will be split away into a wiki page of its own, which can then be edited in the next step. Cleanly adding a new CD altogether is not possible at all, this time due to the incomplete Subversion support of SWiM. SWiM only implements the most basic Subversion commands so far: `update`, `commit`, and `lock`. Other actions like adding and deleting content are possible in the wiki itself but not reflected by its interface to Subversion – which is hacked into the file import/export component instead of being integrated at database level, because the latter would have required a complete overhaul of the design of the underlying IkeWiki system.

4.2 Roadmap

These and other annoyances and missing features (not being able to link to discussion posts, no e-mail notification about discussion posts or page changes, no global search/replace feature across multiple symbols or CDs, to name just a few) are hard to resolve within the existing architecture of SWiM. While some major tasks are definitely within the responsibility of the author, the general usability of the system – besides its adaptation to the mathematical domain – could benefit a lot from improvements to the underlying wiki engine. The development of IkeWiki, which had originally been chosen due to its unique XML and RDF support, has been discontinued, though. On the other hand, its completely reengineered successor KiWi [22] is making good progress.

Therefore, a port of SWiM to KiWi is currently in progress. KiWi's more modular architecture allows for implementing large parts of SWiM not by modifying the core system – as was the case with IkeWiki –, but by providing plugins. New KiWi features of particular interest in the OpenMath setting are a dashboard view giving every user a personalised overview of recent changes at a glance, a service that recommends related content, a facetted search interface, and a concept of transactions that will allow for committing several related changes at once. With the new, improved SWiM system, we will then restart the usability evaluation and work out an accompanying user questionnaire.

A further enhancement planned is replacing the wiki's own database by an integration of Subversion on database level. A database engine capable of

versioning XML documents, particularly mathematical documents, is currently under development in our group [23]. On the user interface end, it is planned to make the OpenMath community benefit from our recent research on active documents. We have implemented interactive services like in-place definition lookup and developed an infrastructure for user-adaptable documents [6].

4.3 Conclusion

We have outlined three CD editing use cases and compared the traditional way of performing them to the new way offered by the SWiM wiki. SWiM clearly excels in these special but common use cases, which has partly been confirmed by the OpenMath CD editors, while still staying compatible with old-style operations going on in the same repository. As SWiM does not yet cover the full CD editing workflow, we presented a roadmap towards its successor, which will rely on a smarter database backend and increase the interactivity of the `http://wiki.openmath.org` site for current and future collaborators and users.

Acknowledgments: The author would like to thank the members of the OpenMath Society, particularly (in alphabetical order) Olga Caprotti, David Carlisle, James Davenport, Paul Libbrecht, Michael Kohlhase, Jan Willem Knopper, and Chris Rowley for giving helpful hints and testing during the design and setup of the wiki, Alberto González Palomo for developing the Sentido formula editor employed by SWiM, and Jakob Ücker for carrying out most of the evaluation work. This work was supported by JEM-Thematic-Network ECP-038208.

References

1. R. Ausbrooks, B. Bos, O. Caprotti, D. Carlisle, G. Chavchanidze, A. Coorg, S. Dalmas, S. Devitt, S. Dooley, M. Hinchcliffe, P. Ion, M. Kohlhase, A. Lazrek, D. Leas, P. Libbrecht, M. Mavrikis, B. Miller, R. Miner, M. Sargent, K. Siegrist, N. Soiffer, S. Watt, and M. Zergaoui. Mathematical Markup Language (MathML) version 3.0. W3C working draft of november 17., World Wide Web Consortium, 2008.
2. J. Carette, L. Dixon, C. Sacerdoti Coen, and S. M. Watt, editors. *MKM/Calculemus 2009 Proceedings*, number 5625 in LNAI. Springer Verlag, 2009. in Press.
3. J. Davenport. A small OpenMath type system. Technical report, The OpenMath Esprit Project, 1999.
4. J. H. Davenport and M. Kohlhase. Unifying Math Ontologies: A tale of two standards. In Carette et al. [2]. in Press.
5. J. H. Davenport and P. Libbrecht. The Freedom to Extend OpenMath and its Utility. *Journal of Mathematics and Computer Science, special issue on Mathematical Knowledge Management*, 2008.
6. J. Giceva, C. Lange, and F. Rabe. Integrating web services into active mathematical documents. In Carette et al. [2], pages 279–293. in Press.
7. M. Kohlhase, C. Lange, C. Müller, N. Müller, and F. Rabe. Notations for active mathematical documents. KWARC Report 2009-1, Jacobs University Bremen, 2009.

8. M. Kohlhase, C. Lange, and F. Rabe. Presenting mathematical content with flexible elisions. In O. Caprotti, M. Kohlhase, and P. Libbrecht, editors, *OpenMath/ JEM Workshop 2007*, 2007.
9. M. Kohlhase, C. Müller, and F. Rabe. Notations for living mathematical documents. In S. Autexier, J. Campbell, J. Rubio, V. Sorge, M. Suzuki, and F. Wiedijk, editors, *Intelligent Computer Mathematics, 9th International Conference, AISC 2008 15th Symposium, Calculemus 2008 7th International Conference, MKM 2008 Birmingham, UK, July 28 - August 1, 2008, Proceedings*, number 5144 in LNAI, pages 504–519. Springer Verlag, 2008.
10. C. Lange. Editing OpenMath content dictionaries with swim. In *3rd JEM Workshop (Joining Educational Mathematics)*, 2008.
11. C. Lange. Mathematical semantic markup in a wiki: The roles of symbols and notations. In C. Lange, S. Schaffert, H. Skaf-Molli, and M. Völkel, editors, *Proceedings of the 3rd Workshop on Semantic Wikis, European Semantic Web Conference 2008*, volume 360 of *CEUR Workshop Proceedings*, Costa Adeje, Tenerife, Spain, June 2008.
12. C. Lange. SWiM – a semantic wiki for mathematical knowledge management. In S. Bechhofer, M. Hauswirth, J. Hoffmann, and M. Koubarakis, editors, *ESWC*, volume 5021 of *Lecture Notes in Computer Science*, pages 832–837. Springer, 2008.
13. C. Lange, U. Bojārs, T. Groza, J. Breslin, and S. Handschuh. Expressing argumentative discussions in social media sites. In J. Breslin, U. Bojārs, A. Passant, and S. Fernández, editors, *Social Data on the Web (SDoW2008), Workshop at the 7th International Semantic Web Conference*, Oct. 2008.
14. C. Lange and A. González Palomo. Easily editing and browsing complex OpenMath markup with SWiM. In P. Libbrecht, editor, *Mathematical User Interfaces Workshop 2008*, 2008.
15. C. Lange, T. Hastrup, and S. Corlosquet. Arguing on issues with mathematical knowledge items in a semantic wiki. In J. Baumeister and M. Atzmüller, editors, *Wissens- und Erfahrungsmanagement LWA (Lernen, Wissensentdeckung und Adaptivität) Conference Proceedings*, volume 448, 2008.
16. J. Marsh, D. Orchard, and D. Veillard. XML inclusions (XInclude) version 1.0 (second edition). W3C Recommendation, World Wide Web Consortium (W3C), Nov. 2006.
17. OpenMath Home. http://www.openmath.org/, seen May 2009.
18. E. Prud'hommeaux and A. Seaborne. SPARQL query language for RDF. W3C Recommendation, World Wide Web Consortium, Jan. 2008.
19. Resource description framework (RDF). http://www.w3.org/RDF/, 2004.
20. RIACA OpenMath products. web page at http://www.riaca.win.tue.nl/projects/openmath/.
21. S. Schaffert. IkeWiki: A semantic wiki for collaborative knowledge management. In *1st International Workshop on Semantic Technologies in Collaborative Applications STICA 06, Manchester, UK*, June 2006.
22. S. Schaffert, J. Eder, S. Grünwald, T. Kurz, M. Radulescu, R. Sint, and S. Stroka. KiWi – a platform for semantic social software. In C. Lange, S. Schaffert, H. Skaf-Molli, and M. Völkel, editors, *Proceedings of the 4th Workshop on Semantic Wikis, European Semantic Web Conference 2009*, Hersonissos, Greece, June 2009. in press.
23. TNTBase Project, seen December 2008. available at https://trac.mathweb.org/tntbase/.
24. The Trac Project. http://trac.edgewall.org/, 2008.

The `order1` Content Dictionary

Sylla Lesseni[1] and Dan Roozemond[2]

[1] Fakultät II - Institut für Mathematik, Technische Universität Berlin,
`lesseni@math.tu-berlin.de`
[2] Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, Netherlands, `d.a.roozemond@tue.nl`

Brief Description

We propose a content dictionary for orders of number fields written for SCIEnce project [3]. Reference textbooks on the topic are for example by Pohst and Zassenhous [2] and Cohen [1]. The proposed `order1` content dictionary contains symbols for the basic functions and constructors in this field.

This CD is the first one of the CDs we hope to develop to deal with number fields. Using these tools and methods, one can handle the absolute number fields (defined over the rational field $\mathbb{Q}$) and relative number fields (defined over a number field); these different notions were not stated anywhere in the existing CDs.

The accompanying `.ocd` and `.html` files contain the relevant descriptions and some examples.

(Note added following referee comments: Probably `numfield1` would be a more suitable name)

References

1. H. Cohen, *A course in Computational Algebraic Number Theory*. Berlin, Springer-Verlag (1993).
2. M. Pohst and H. Zassenhaus, *Algorithmic Algebraic Number Theory*. Cambridge Univ. Press, 1989.
3. Symbolic Computation Infrastructure for Europe. `http://www.symbolic-computation.org/`

This work was carried out under the SCIEnce project (Symbolic Computation Infrastructure for Europe), an *Integrated Infrastructure Initiative* funded by the European Commission.

OpenMath Content Dictionary: order1

Canonical URL:
http://www.win.tue.nl/SCIEnce/cds/order1.ocd
CD File:
order1.ocd
CD as XML Encoded OpenMath:
order1.omcd
Defines:
algebraic_integer, algebraic_number, is_Dedekind, is_maximal_order, is_nonzero_divisor, is_principal_ideal_domain, maximal_order, number_field, order, primitive_element, ring_integers
Date:
2009-06-22
Version:
1 (Revision 6)
Review Date:
Status:
experimental
Uses CD:
alg1, arith1, fieldname1, logic1, nums1, polyd1, quant1, relation1, ring1, ringname1, set1, setname1

This CD defines the basic functions and constructors for orders of number fields. Written by S. Lesseni (lesseni@math.tu-berlin.de).

```
A CD of basic functions for orders of number fields written for SCIEnce project. Note
The reference textbooks are:
(1) M. Pohst and H. Zassenhaus, Algorithmic Algebraic Number Theory, Cambridge
Univ. Press, 1989.
(2) H. Cohen, A course in Computational Algebraic Number Theory. Berlin, Springer-Ver
```

is_Dedekind

Role:
application

Description:
This symbol represents a unary boolean function. The argument should be a ring R. When evaluated on R, the function returns true if R is a Dedekind ring and false otherwise. Note that a ring R is a Dedekind ring if it is Noetherian, integrally closed (so integral) and such that every non-zero prime ideal is maximal.

Commented Mathematical property (CMP):
if R is a Dedekind ring and a subring of the rational field Q then R = Z.

Formal Mathematical property (FMP):
xml prefix mathml

is_Dedekind (R) ∧ is_subring (Q , R) ⇒ R = Z

Example:

if the ring (R,+,0,-,*,1) is a principal ideal domain then (R,+,0,-,*,1) is a Dedekind ring.

xml prefix mathml

∀ R , plus , zero , minus , times , one . is_principal_ideal_domain (ring (R , plus , zero , minus , times , one)) ⇒ is_Dedekind (ring (R , plus , zero , minus , times , one))

Signatures:

sts

[Next: is_nonzero_divisor] [Last: primitive_element] [Top]

---

is_nonzero_divisor

Role:

application

Description:

This symbol represents a boolean binary function. The first argument is a ring R, the second is an element b of R. When evaluated on R and b, the function returns true if b is a nonzero divisor in R.

Commented Mathematical property (CMP):

if x is a non-zero divisor in the ring R then x is in R and x is different from zero and for all non-zero y in R, x*y is different from zero or y*x is different from zero.

Formal Mathematical property (FMP):

xml prefix mathml

is_nonzero_divisor (R , x) ⇒ x ∈ carrier (R) ∧ x ≠ zero (R) ∧ (∀ y . y ∈ carrier (R) ∧ y ≠ zero (R) ⇒ ((multiplication (R)) (x , y) ≠ zero (R)) ∨ ((multiplication (R)) (y , x) ≠ zero (R)) ⇒)

Example:

xml prefix mathml

is_nonzero_divisor (Z , 5)

Signatures:

sts

[Next: is_principal_ideal_domain] [Previous: is_Dedekind] [Top]

---

is_principal_ideal_domain

Role:

application

Description:

The unary boolean function whose value is true if and only if the argument is a principal ideal domain. R is a principal ideal domain if R is a commutative ring without zero divisors and if every ideal of R is a principal ideal.

Commented Mathematical property (CMP):

is_principal_ideal_domain(R) then for all a, b in R a*b=b*a and a different from zero in R then a is a

non-zero divisor in R and I an ideal of R then there exists x in R such that I is the principal ideal generated by x in R.

Formal Mathematical property (FMP):

xml prefix mathml

`order1` *CD*

is_principal_ideal_domain (R) ⇒ ∀ a , b . a ∈ carrier (R) ∧ b ∈ carrier (R) ⇒ (multiplication (R)) (a , b) = (multiplication (R)) (b , a) ∧ (a ∈ carrier (R) ⇒ is_nonzero_divisor (R , a)) ∧ (is_ideal (carrier (R) , I) ⇒ ∃ x . x ∈ carrier (R) ⇒ I = principal_ideal (R , x))

Example:

xml prefix mathml

is_principal_ideal_domain (Z)

Signatures:

sts

[Next: order] [Previous: is_nonzero_divisor] [Top]

---

order

Role:

application

Description:

This is a binary function, whose first argument is a Dedekind ring R and the second is a polynomial f. When applied to R and f, it returns an order of f over the ring of polynomial of R: it is a ring A containing R, which is finitely generated R-module with no nilpotent non-zero ideal and as a R-module it is torsion-free. Note that the result is not unique. Also this function allows to compute an order of a polynomial over another polynomial ring. The idea behind this computation is to coerce f into the polynomial ring of R and then compute the order.

Example:

xml prefix mathml

order (Z , DMP (poly_ring_d (Z , 1) , SDMP (term (1 , 2) , term (3 , 0))))

Signatures:

sts

[Next: maximal_order] [Previous: is_principal_ideal_domain] [Top]

---

maximal_order

Role:

application

Description:

This is a binary function, whose first argument is a Dedekind ring R and the second is a polynomial f. When applied to R and f, it returns the maximal order A among the orders of f (over the polynomial ring of R) in the quotient field of A. Note that the result is unique.

Example:

xml prefix mathml

maximal_order (Z , DMP (poly_ring_d (Z , 1) , SDMP (term (1 , 2) , term (3 , 0))))

Signatures:
sts

[Next: is_maximal_order] [Previous: order] [Top]

is_maximal_order

Role:
application

Description:
The unary boolean function whose value is true if and only if the argument is a maximal order.

Example:

xml prefix mathml

is_maximal_order (maximal_order (Z , DMP (poly_ring_d (Z , 1) , SDMP (term (1 , 2) , term (3 , 0)))))

Signatures:
sts

[Next: algebraic_integer] [Previous: maximal_order] [Top]

algebraic_integer

Role:
application

Description:
This is a binary function. The first argument is an order O. The second argument should be a list L of elements of the Dedekind ring R, such that O is an order over the polynomial ring of R (cf. order). The length of L should be equal to the degree n of the polynomial generating the order O. When applied to O and L, it represents the element L[0] + L[1] b + L[2] b^2 + ... + L[n-1] b^(n-1) of O, where b stands for a primitive element of O.

Example:

xml prefix mathml

algebraic_integer (order (Z , DMP (poly_ring_d (Z , 1) , SDMP (term (1 , 2) , term (3 , 0)))) , (7 , 2))

Signatures:
sts

[Next: number_field] [Previous: is_maximal_order] [Top]

number_field

Role:
application

Description:
This symbol is a constructor for number fields. It takes two arguments in the following order: a ring R and a monic irreducible univariate polynomial f. If the ring R is Z (or Q), it returns the absolute number field. Otherwise it returns the relative number field over the number field whose ring of integers is R. This symbol is intended to be used in upcoming CDs for e.g. describing discriminants of number fields, or Galois groups, unit groups, class groups, regulators, etc.; all useful number theoretical notions.

Commented Mathematical property (CMP):
number_field(Z,x^2+1)

Formal Mathematical property (FMP):
xml prefix mathml

number_field (Z , DMP (poly_ring_d (Z , 1) , SDMP (term (2 , 1) , term (0 , 1))))

Example:
xml prefix mathml

number_field (ring_integers (number_field (Z , DMP (poly_ring_d (Z , 1) , SDMP (term (1 , 2) , term (2 , 0))))) , DMP (poly_ring_d (Z , 1) , SDMP (term (1 , 2) , term (3 , 0))))

Signatures:
sts

[Next: algebraic_number] [Previous: algebraic_integer] [Top]

---

algebraic_number

Role:
application

Description:
This is a binary function. The first argument is a number field F. The second argument should be a list L of elements of Q in case of an absolute number field F. Otherwise the second argument is a list L of elements of the number field whose ring of integers is the ring R over which F is defined (cf. number_field). The length of the list L should be equal to the degree n of F. When applied to F and L, it represents the element L[0] + L[1] b + L[2] b^2 + ... + L[n-1] ^(b-1) of F, where b stands for a primitive element of F.

Example:
xml prefix mathml

algebraic_number (order (Z , DMP (poly_ring_d (Z , 1) , SDMP (term (1 , 2) , term (1 , 0))) , (123 , 0)))

Signatures:
sts

[Next: ring_integers] [Previous: number_field] [Top]

---

ring_integers

Role:
application

Description:
This is a unary function, whose argument is a number field K. When applied to K, it returns the ring of integers of K. It is the Dedekind ring of K.

Commented Mathematical property (CMP):
if A is the ring of integers of the number field K then A is a subring of K and A is a Dedekind ring.

Formal Mathematical property (FMP):
xml prefix mathml

A = ring_integers (K) ⇒ is_subring (K , A) ∧ is_Dedekind (A)

Example:
xml prefix mathml

ring_integers (number_field (Z , DMP (poly_ring_d (Z , 1) , SDMP (term (1 , 2) , term (2 , 0)))))

Signatures:
sts

[Next: primitive_element] [Previous: algebraic_number] [Top]

---

primitive_element

Role:
application

Description:
This is a unary function, whose argument is a number field K. It returns a primitive element of K. Note that the result is not unique.

Example:
xml prefix mathml

primitive_element (number_field (Z , DMP (poly_ring_d (Z , 1) , SDMP (term (1 , 2) , term (2 , 0)))))

Signatures:
sts

[First: is_Dedekind] [Previous: ring_integers] [Top]

The matrix1 Content Dictionary

Sebastian Freundt[1], Peter Horn[2], and Dan Roozemond[3]

[1] Fakultät II - Institut für Mathematik, Technische Universität Berlin,
freundt@math.tu-berlin.de
[2] Universität Kassel, Heinrich Plett Straße 40, 34132 Kassel,
horn@math.uni-kassel.de,
[3] Technical Universiteit Eindhoven, Den Dolech 2, Postbus 513, 5600 MB Eindhoven,
d.a.roozemond@tue.nl

Brief Description

During the development of the Symbolic Computation Software Composability Protocol in the SCIEnce project [1, 2] we observed that the OpenMath support for matrices, in particular described in the content dictionaries `linalg1` and `linalg2`, was not quite extended enough for us.
We therefore propose the `matrix1` content dictionary, which adds in particular:

- the possibility to specify the entry domain of the entries of a matrix beforehand. This enables for example more efficient transmission of matrices over finite fields, because in `linalg1` and `linalg2` it is required to specify the ground field with every entry.
- the possibility to efficiently transmit sparse matrices.
- very verbose syntax (cf `row_dimension` and `column_dimension`) that enables easy parsing.

Especially for sparse matrices the speed up is, as expected, enormous. For instance for permutations matrices of size 1000, the size of the OpenMath XML object is 12MB when using `linalg2` compared to 90KB when using the proposed `matrix1` content dictionary. (In the binary representation the difference is of a similar order of magnitude: 4MB vs 35KB).

The accompanying `.ocd` and `.html` files contain the relevant descriptions and some examples.

References

1. S. Freundt, P. Horn, A. Konovalov, S. Linton and D. Roozemond. Symbolic Computation Software Composability. In *Intelligent Computer Mathematics, AISC/Calculemus/MKM 2008 proceedings*, Lecture Notes in Computer Science 5144/2008, Springer, p.285-295.
2. Symbolic Computation Infrastructure for Europe. `http://www.symbolic-computation.org/`

This work was carried out under the SCIEnce project (Symbolic Computation Infrastructure for Europe), an *Integrated Infrastructure Initiative* funded by the European Commission.

OpenMath Content Dictionary: matrix1

Canonical URL:
http://www.win.tue.nl/SCIEnce/cds/matrix1.ocd
CD File:
matrix1.ocd
CD as XML Encoded OpenMath:
matrix1.omcd
Defines:
banded, block, column_dimension, dense, diagonal, entry_domain, lower_band, matrix, matrix_domain, row_dimension, sparse, sparse_entry, upper_band
Date:
2009-06-22
Version:
0 (Revision 4)
Review Date:
Status:
experimental
Uses CD:
alg1, ringname1, linalg2, linalg3

This CD holds a collection of matrix constructors over arbitrary rings.

---

entry_domain

Description:
This symbol is a unary function, whose argument should be a ring r. When applied to r, it represents the matrix-algebra ground domain (MAD).

Example:
xml prefix mathml

entry_domain (Z)

Signatures:
sts

[Next: matrix_domain] [Last: lower_band] [Top]

---

matrix_domain

Description:
This symbol is a ternary function, whose first argument should be a matrix1.entry_domain application. The second and third arguments must be matrix1.row_dimension and matrix1.column_dimension. When applied to these arguments this `creates' the domain of linear mappings between modules of specified dimensions over a common ground domain, conveniently represented by matrices.

Example:

xml prefix mathml

matrix_domain (entry_domain (Z) , row_dimension (12) , column_dimension (10))

82 matrix1 *CD*

Signatures:

sts

[Next: row_dimension] [Previous: entry_domain] [Top]

---

row_dimension

Description:

This symbol is a unary function whose first argument must be either a non-negative OpenMath integer or nums1.infinity. When applied this creates an object that denotes the dimension of the codomain of the linear mapping represented by the matrix.

Signatures:

sts

[Next: column_dimension] [Previous: matrix_domain] [Top]

---

column_dimension

Description:

This symbol is a unary function whose first argument must be either a non-negative OpenMath integer or nums1.infinity. When applied this creates an object that denotes the dimension of the domain of the linear mapping represented by the matrix.

Signatures:

sts

[Next: matrix] [Previous: row_dimension] [Top]

---

matrix

Description:

This symbol is a binary function whose first argument must be a matrix algebra constructor and the second argument can be any of the below matrix entry constructors. Additionally it is possible to use the matrix constructors of the linalg2 or linalg3 CDs.

Example:

xml prefix mathml

Todo

Signatures:

sts

[Next: dense] [Previous: column_dimension] [Top]

---

dense

Description:

This symbol is an $(m \cdot n)$-ary function whose arguments specify the entries of the matrix, where m is the dimension of the codomain and n is the dimension of the domain. The matrix (or block) must be filled row-wise, that is the first argument denotes the entry in row 1, column 1 of the matrix (or block), the second argument denotes the entry at row 1, column 2, and so forth. The number of arguments MUST match the dimensions of either the matrix algebra or the surrounding block (see below).

Example:

xml prefix mathml

matrix (matrix_domain (entry_domain (Z) , row_dimension (3) , column_dimension (3)) , dense (1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9))

Signatures:

sts

[Next: sparse] [Previous: matrix] [Top]

---

sparse

Description:

The constructor for sparse matrices without any indication of dimension or domain for the coefficients. Its arguments are just matrix1.sparse_entrys. Attention: No two matrix1.sparse_entrys must specify the same location.

Signatures:

sts

[Next: sparse_entry] [Previous: dense] [Top]

---

sparse_entry

Description:

This symbol denotes a ternary function whose first two arguments specify the location of an entry inside the matrix, and whose final argument is the entry itself. The entry MUST be either from the specified ground domain directly, or be a diagonal constructor as described below, a block constructor as described below, or a banded constructor as described below. In the block case, the dimensions of the block MUST NOT exceed the total dimensions of the matrix algebra. In the diagonal case, the dimension of the diagonal MUST NOT exceed the total dimensions of the matrix algebra.

Example:

xml prefix mathml

matrix (matrix_domain (entry_domain (Q) , row_dimension (3) , column_dimension (3)) , sparse (sparse_entry (1 , 2 , 12) , sparse_entry (2 , 1 , 21) , sparse_entry (3 , 3 , 33)))

Signatures:

sts

[Next: diagonal] [Previous: sparse] [Top]

---

diagonal

Description:

This symbol is an m-ary function whose arguments specify the entries of a (generalised) matrix diagonal. The diagonal must be filled from top-left to bottom-right. That is: the first argument represents the entry in row 1, column 1 of the matrix, the second argument denotes the entry at row 2, column 2, and so forth. If used inside a sparse_entry object at location (i, j), the first entry is offset accordingly. If not used inside a sparse_entry object, the number of arguments MUST match the dimensions of either the matrix algebra (the smaller of m and n). If used inside a sparse_entry object, the number of entries MUST NOT exceed the total matrix dimensions.

Example:

xml prefix mathml

matrix (matrix_domain (entry_domain (C) , row_dimension (3) , column_dimension (3)) , diagonal (1 + i , 2 + 2 i , 3 + 3 i))

Signatures:

sts

[Next: block] [Previous: sparse_entry] [Top]

---

block

Description:

This symbol is like the matrix constructor as described above, but intended for use inside matrix to form ``submatrices''. The symbol takes at least two arguments: a column_dimension and a row_dimension object which denote the total extent of the block.

Example:

xml prefix mathml

matrix (matrix_domain (entry_domain (Q) , row_dimension (30) , column_dimension (30)) , sparse (sparse_entry (10 , 20 , block (row_dimension (2) , column_dimension (2) , dense (11 , 12 , 21 , 22)))))

Example:

xml prefix mathml

matrix (matrix_domain (entry_domain (Z) , row_dimension (1000000) , column_dimension (1000000)) , sparse (sparse_entry (24800 , 26133 , block (row_dimension (99999) , column_dimension (99999) , sparse (sparse_entry (4 , 15 , 0))))))

Signatures:

sts

[Next: banded] [Previous: diagonal] [Top]

---

banded

Description:

This symbol is a constructor for banded matrices. It takes at least 2 arguments, the first of which being the number of upper bands and the second being the number of lower bands. Amongst the further arguments

you can specify AT MOST one matrix1.diagonal object. You MUST exactly as many matrix1.upper_band objects as you specified upper bands, and you MUST specify as many matrix1.lower_band objects as you specified lower bands. This symbol facilitates the use of blas based systems which expect to know the bands structure upfront.

Example:

xml | prefix | mathml

matrix (matrix_domain (entry_domain (Zm (7)) , row_dimension (3) , column_dimension (3)) , banded (1 , 1 , diagonal (111 , 222 , 333) , upper_band (1 , diagonal (4 , 5)) , lower_band (1 , diagonal (1 , 2))))

Signatures:

sts

[Next: upper_band] [Previous: block] [Top]

---

upper_band

Description:

This symbol is a binary function whose first argument is a non-negative OpenMath integer which denotes the index of the upper band which is specified in the second argument. Hereby the first upper band is the one immediately above the main (generalised) diagonal, its starting coordinates relative to the top-left of the matrix thus are (1, 2).

Signatures:

sts

[Next: lower_band] [Previous: banded] [Top]

---

lower_band

Description:

This symbol is a binary function whose first argument is a non-negative OpenMath integer which denotes the index of the lower band which is specified in the second argument. Hereby the first lower band is the one immediately below the main (generalised) diagonal, its starting coordinates relative to the top-left of the matrix thus are (2, 1).

Signatures:

sts

[First: entry_domain] [Previous: upper_band] [Top]

The polynomial4 Content Dictionary

Sebastian Freundt[1], Peter Horn[2], and Dan Roozemond[3]

[1] Fakultät II - Institut für Mathematik, Technische Universität Berlin,
freundt@math.tu-berlin.de
[2] Universität Kassel, Heinrich Plett Straße 40, 34132 Kassel,
horn@math.uni-kassel.de,
[3] Technical Universiteit Eindhoven, Den Dolech 2, Postbus 513, 5600 MB Eindhoven,
d.a.roozemond@tue.nl

Brief Description

During the development of the Symbolic Computation Software Composability Protocol in the SCIEnce project [1, 2] we observed that the OpenMath support for polynomials, in particular considering factorization, could be improved.

We therefore propose the polynomial4 content dictionary, which is intended to cooperate with the existing polynomial content dictionaries. While writing this content dictionary we had in particular the polyd family in mind. It adds for example:

- symbols to hold both the quotient and the remainder of a polynomial division. This is convenient for example when asking a computer algebra system to perform such a division, as quotient and remainder are often computed simultaneously, and the user is likely interested in both.
- extended support for factorization of polynomials. For example, this content dictionary provides symbols to indicate results that may or may not be irreducible, complete and incomplete factorisations, and multiple factorisations in case the polynomial algebra is not a unique factorisation domain.

The accompanying .ocd and .html files contain the relevant descriptions and some examples.

References

1. S. Freundt, P. Horn, A. Konovalov, S. Linton and D. Roozemond. Symbolic Computation Software Composability. In *Intelligent Computer Mathematics, AISC/Calculemus/MKM 2008 proceedings*, Lecture Notes in Computer Science 5144/2008, Springer, p.285-295.
2. Symbolic Computation Infrastructure for Europe. http://www.symbolic-computation.org/

This work was carried out under the SCIEnce project (Symbolic Computation Infrastructure for Europe), an *Integrated Infrastructure Initiative* funded by the European Commission.

OpenMath Content Dictionary: polynomial4

Canonical URL:
http://www.win.tue.nl/SCIEnce/cds/polynomial4.ocd
CD File:
polynomial4.ocd
CD as XML Encoded OpenMath:
polynomial4.omcd
Defines:
definitely_irreducible, divide, factor, factorisations, factorisations_complete, factorisations_incomplete, factorise, factors, ground_ring_injected, multiplicity, possibly_reducible, quotient, quotient_remainder, remainder
Date:
2009-06-22
Version:
0 (Revision 4)
Review Date:
Status:
experimental
Uses CD:
alg1, polyd1

This CD holds a collection of for some operations of polynomials over rings. The data structures for polynomials can be arithmetic expressions, for instance using the ring1.expression symbol, or DMP as in the CD polyd1.

---

factorise

Description:
This symbol is a unary function, whose argument should be a polynomial f. When applied to f, it represents a list of factors of f. Cf. polynomial4.factorisations for a description of the expected reply.

Example:
xml prefix mathml

factorise (DMP (poly_ring_d_named (Z , X) , SDMP (term (1 , 2) , term (-1 , 0))))

Signatures:
sts

[Next: factorisations] [Last: remainder] [Top]

---

factorisations

Description:
This symbol may be used in the reply of polynomial4.factorise and takes at least 1 argument. The first

argument is one of polynomial4.factorisations_complete to indicate that the following list of polynomial4.factors cells covers all possible factorisations. The counterpart would be polynomial4.factorisations_possibly_incomplete to indicate that the following list of factorisations are some of possibly many more factorisations. Note: If the polynomial algebra is a UFD (unique factorisation domain) the uniqueness can be underpinned by giving exactly one polynomial4.factors cell and using the symbol polynomial4.factorisations_complete here. The rest of the arguments are polynomial4.factors cells, each of which being a possible factorisation. Using the call of polynomial4.factorise above we might obtain:

Example:

xml prefix mathml

factorisations (factorisations_complete , factors (definitely_irreducible , poly_ring_d_named (Z , X) , 1 , factor (DMP (poly_ring_d_named (Z , X) , SDMP (term (1 , 1) , term (-1 , 0))) , multiplicity (1)) , factor (DMP (poly_ring_d_named (Z , X) , SDMP (term (1 , 1) , term (1 , 0))) , multiplicity (1))))

Signatures:

sts

[Next: factors] [Previous: factorise] [Top]

---

factors

Description:

This symbol is used in the reply of polynomial4.factorise and takes at least 2 arguments. Note this symbol may also be used in a polynomial4.factorisations cell. The first argument is one of polynomial4.definitely_irreducible or polynomial4.possibly_reducible and specifies whether the computed factorisation is known to be irreducible or if the irreducibility of some of the factors is not guaranteed. Note: This symbol is mandatory even if the factors themselves (see polynomial4.factor) can carry that information, this is simply to connive at computer algebra systems that cannot figure out which of the factors is the possibly reducible one. Generally this slot must be polynomial4.possibly_reducible if at least one of the factors is possibly reducible. The second argument contains a polyd1.poly_ring_d or polyd1.poly_ring_d_named cell, as specified in e.g. polyd or polyd1 to indicate the underlying polynomial algebra. The third argument is a symbol polynomial4.common_coefficient and denotes the common coefficient of the factorisation. Note: In case the ground ring itself is regarded as being injected into the polynomial algebra, or the factorisation is normalised, this field may be used to specify the unit giving the normalisation. Furthermore, the cell comprises polynomial4.factor cells which in turn represent the factors of the polynomial in a factorisation along with their multiplicities. Using the call of polynomial4.factorise above we might obtain:

Example:

xml prefix mathml

factors (definitely_irreducible , poly_ring_d_named (Z , X) , 1 , factor (DMP (poly_ring_d_named (Z , X) , SDMP (term (1 , 1) , term (-1 , 0))) , multiplicity (1)) , factor (DMP (poly_ring_d_named (Z , X) , SDMP (term (1 , 1) , term (1 , 0))) , multiplicity (1)))

Signatures:

sts

[Next: factor] [Previous: factorisations] [Top]

---

factor

Description:

A symbol which represents one factor of a factorisation, it takes at least 2 arguments, the first of which being the factor polynomial, e.g. a polyd1.DMP, and the second being its multiplicity specified as an integer >= 1. Optionally, the third argument is one of polynomial4.definitely_irreducible, polynomial4.possibly_reducible to indicate whether or not the given factor is guaranteed to be irreducible. Furthermore, this symbol may contain polynomial4.ground_ring_injected to indicate that the ground ring is considered to be embedded in the polynomial algebra and hence the factor is actually the factorisation of a polynomial coefficient.

Signatures:

sts

[Next: multiplicity] [Previous: factors] [Top]

---

multiplicity

Description:

A symbol which represents the multiplicity of a factor in a factorisation and takes exactly one argument which must be a positive integer.

Signatures:

sts

[Next: factorisations_complete] [Previous: factor] [Top]

---

factorisations_complete

Description:

A symbol to indicate that a given list of factorisations of a polynomial covers in fact all possible factorisations.

Signatures:

sts

[Next: factorisations_incomplete] [Previous: multiplicity] [Top]

---

factorisations_incomplete

Description:

A symbol to indicate that a given list of factorisations is an assortment of all possible factorisations.

Signatures:

sts

[Next: definitely_irreducible] [Previous: factorisations_complete] [Top]

---

definitely_irreducible

Description:

A symbol which denotes that a factor of the factorisation is definitely irreducible.

Signatures:

sts 90 polynomial4 *CD*

[Next: possibly_reducible] [Previous: factorisations_incomplete] [Top]

---

possibly_reducible

Description:

A symbol which denotes that the irreducibility of a factor of the factorisation is not guaranteed.

Signatures:

sts

[Next: ground_ring_injected] [Previous: definitely_irreducible] [Top]

---

ground_ring_injected

Description:

A symbol which denotes that the ground ring of a polynomial algebra is considered to be part of the latter. This is used in the polynomial4.factor symbol to indicate that the factor is part of the factorisation of the common coefficient.

Signatures:

sts

[Next: divide] [Previous: possibly_reducible] [Top]

---

divide

Description:

This symbol is a binary function whose arguments are polynomials f and g which must be defined over the same ground domain. When applied to f and g, it represents the quotient arising from dividing f by g and the remainder h such that h is congruent f modulo g. The result is gathered in a polynomial4.quotient_remainder cell. Hint: We consider named polynomial rings, i.e. the indeterminate is explicitly specified by a named variable, different once the variable names differ. That is, a polynomial in Z[X] cannot be divided by a polynomial in Z[Y] a priori. However, we leave it up to the implementor to handle this differently, though we strongly encourage implementors to return a polynomial in an anonymous indeterminate (using e.g. polyd1.poly_ring_d rather than polyd1.poly_ring_d_named).

Example:

xml prefix mathml

divide (DMP (poly_ring_d (Z , 1) , SDMP (term (1 , 5) , term (2 , 3) , term (1 , 0))) , DMP (poly_ring_d (Z , 1) , SDMP (term (1 , 2) , term (1 , 0))))

Signatures:

sts

[Next: quotient_remainder] [Previous: ground_ring_injected] [Top]

quotient_remainder

Freundt et al. 91

Description:

This symbol is a container for the result of polynomial4.divide. It takes 2 arguments in unspecified order, polynomial4.quotient and polynomial4.remainder. Using the above polynomial4.divide call we may obtain:

Example:

xml | prefix | mathml

quotient_remainder (quotient (DMP (poly_ring_d (Z , 1) , SDMP (term (1 , 3) , term (1 , 1)))) , remainder (DMP (poly_ring_d (Z , 1) , SDMP (term (-1 , 1) , term (1 , 0)))))

Signatures:

sts

[Next: quotient] [Previous: divide] [Top]

quotient

Description:

This symbol contains the quotient of polynomial4.divide. Cf. polynomial4.quotient_remainder for an example.

Signatures:

sts

[Next: remainder] [Previous: quotient_remainder] [Top]

remainder

Description:

This symbol contains the remainder of polynomial4.divide. Cf. polynomial4.quotient_remainder for an example.

Signatures:

sts

[First: factorise] [Previous: quotient] [Top]

Home Overview Documents Content Dictionaries Software & Tools The OpenMath Society OpenMath Projects OpenMath Discussion Lists OpenMath Meetings Links

The scscp1 and scscp2 Content Dictionaries

Sebastian Freundt[1], Peter Horn[2], Alexander Konovalov[3], Sylla Lesseni[1], Steve Linton[2], and Dan Roozemond[3]

[1] Fakultät II - Institut für Mathematik, Technische Universität Berlin, Berlin, Germany, {freundt|lesseni}@math.tu-berlin.de
[2] Fachbereich Mathematik, Universität Kassel, Kassel, Germany, hornp@mathematik.uni-kassel.de
[3] School of Computer Science, University of St Andrews, Scotland, {alexk|sal}@mcs.st-and.ac.uk
[4] Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, Netherlands, d.a.roozemond@tue.nl

Brief Description

These content dictionaries are a result of one of the activities of the European project "SCIEnce" [2]. The main aim of this project is to allow unified communication between different computer algebra systems (CASes) or different instances of one CAS. This may involve one or more computers, clusters, and even grids.

When designing such a uniform communication interface for CASes, the first problem that needs to be solved is how to transport the mathematical objects from one system to another. Here, the obvious choice for us was OpenMath.

To assist the communication between the various CASes, we have developed a protocol called "Symbolic Computation Software Composability Protocol", abbreviated SCSCP [1, 3]. This protocol does not only enable the computation of simple commands in a different system or on a different machine, but it will also serve as a means of conveying constituents of larger, more complex, computations.

The protocol is XML-based; in particular, the protocol messages are in the OpenMath language itself. The OpenMath symbols used appear in two content dictionaries: `scscp1` and `scscp2`. The accompanying `.ocd` and `.html` files contain the relevant descriptions and some examples.

References

1. S. Freundt, P. Horn, A. Konovalov, S. Linton and D. Roozemond. Symbolic Computation Software Composability. In *Intelligent Computer Mathematics, AISC/Calculemus/MKM 2008 proceedings*, Lecture Notes in Computer Science 5144/2008, Springer, p.285-295.
2. Symbolic Computation Infrastructure for Europe. http://www.symbolic-computation.org/
3. S. Freundt, P. Horn, A. Konovalov, S. Linton, D. Roozemond, Symbolic Computation Software Composability Protocol (SCSCP) specification, Version 1.3, 2009. http://www.symbolic-computation.org/scscp/

OpenMath Content Dictionary: scscp1

Canonical URL:
http://www.win.tue.nl/SCIEnce/cds/scscp1.ocd
CD File:
scscp1.ocd
CD as XML Encoded OpenMath:
scscp1.omcd
Defines:
call_id, error_memory, error_runtime, error_system_specific, info_memory, info_message, info_runtime, option_debuglevel, option_max_memory, option_min_memory, option_return_cookie, option_return_nothing, option_return_object, option_runtime, procedure_call, procedure_completed, procedure_terminated
Date:
2009-06-22
Version:
1 (Revision 12)
Review Date:
Status:
experimental

This CD defines symbols for the description of the management of mathematical queries. In particular, it is used by the SCIEnce project [SCIEnce website] in the communication between a web service (i.e. computer algebra system, proof checker, etc) and a client. SCSCP is an abbreviation for "Symbolic Computation Software Composability Protocol" [SCSCP specification].

This version of the Content Dictionary agrees with version 1.3 of the SCSCP protocol.

procedure_call

Description:
The actual procedure call. Its only argument is an OpenMath Application, whose head symbol describes the procedure to be called, and whose arguments are the arguments to the procedure.

Example:
xml prefix mathml

procedure_call (GroupIdentificationService (group (permutation (2 , 3 , 1) , permutation (1 , 2 , 4 , 3))))

Signatures:
sts

[Next: procedure_completed] [Last: error_system_specific] [Top]

procedure_completed

Description:

The result of a successful computation. Should come along with a call_id and, possibly, some extra information.

Example:

xml | prefix | mathml

procedure_completed (2692574850823428107600 9)

Instead of the result, we may return a reference to the result, as follows:

Example:

xml | prefix | mathml

procedure_completed ( )

Signatures:

sts

[Next: procedure_terminated] [Previous: procedure_call] [Top]

---

procedure_terminated

Description:

The result of a failed computation. Should come along with a call_id, an error description, and possibly some extra information.

Example:

xml | prefix | mathml

procedure_terminated (error_system_specific Segmentation fault)

Signatures:

sts

[Next: call_id] [Previous: procedure_completed] [Top]

---

call_id

Description:

Uniquely identifies a procedure call. Used in subsequent communication, so the parties know which call they are talking about.

Example:

xml | prefix | mathml

Signatures:

sts

[Next: option_max_memory] [Previous: procedure_terminated] [Top]

---

option_max_memory

Description:

An option, to be given along with a procedure call, describing the maximum amount of memory (in bytes) the system should spend on this call.

Example:

xml prefix mathml

Signatures:

sts

[Next: option_min_memory] [Previous: call_id] [Top]

---

option_min_memory

Description:

An option, to be given along with a procedure call, describing the minimum amount of memory (in bytes) the system should be able to spend on this call. The idea is that in certain cases we know in advance that we will need a large amount of memory. If the system will never be able to provide that, it would be a waste of time and resources to even start the computation.

Example:

xml prefix mathml

Signatures:

sts

[Next: option_runtime] [Previous: option_max_memory] [Top]

---

option_runtime

Description:

An option, to be given along with a procedure call, describing the maximum amount of time (in milliseconds) the system should spend on this call.

Example:

xml prefix mathml

Signatures:

sts

[Next: option_debuglevel] [Previous: option_min_memory] [Top]

---

option_debuglevel

Description:

An option, to be given along with a procedure call, describing the amount of debug information the client is interested in. Should be an integer.

Example:

xml prefix mathml

Signatures:

sts

[Next: option_return_cookie] [Previous: option_runtime] [Top]

option_return_cookie

Description:

An option, to be given along with a procedure call, indicating that the client would like to have a cookie (i.e. a reference to an OpenMath object residing somewhere) as return value.

Example:

xml prefix mathml

The reply from the server should then look like:

Example:

xml prefix mathml

procedure_completed ( )

Signatures:

sts

[Next: option_return_object] [Previous: option_debuglevel] [Top]

option_return_object

Description:

An option, to be given along with a procedure call, indicating that the client would like to have the actual OpenMath object as return value.

Example:

xml prefix mathml

The reply from the server should then look like:

Example:

xml prefix mathml

procedure_completed (42)

Signatures:

sts

[Next: option_return_nothing] [Previous: option_return_cookie] [Top]

option_return_nothing

Description:

An option, to be given along with a procedure call, indicating that the client expects no return value.

Example:

xml prefix mathml

The reply from the server may then look like:

Example:

xml prefix mathml

procedure_completed ( )

Signatures:

sts

[Next: info_memory] [Previous: option_return_object] [Top]

---

info_memory

Description:

A piece of information from the system, to be used along with a procedure_completed or procedure_terminated message, describing how much memory was spent on the calculation. It should be in bytes, denoted using an OMI.

Example:

xml prefix mathml

Signatures:

sts

[Next: info_runtime] [Previous: option_return_nothing] [Top]

---

info_runtime

Description:

A piece of information from the system, to be used along with a procedure_completed or procedure_terminated message, describing how much cputime was spent on the calculation. It should be in milliseconds, denoted using an OMI.

Example:

xml prefix mathml

Signatures:

sts

[Next: info_message] [Previous: info_memory] [Top]

---

info_message

Description:

A piece of information from the server, to be used along with a procedure_completed or procedure_terminated message, giving some additional information. The client may choose to present this

information to its user. The argument is an OMSTR.

Example:

xml prefix mathml

Signatures:

sts

[Next: error_memory] [Previous: info_runtime] [Top]

---

error_memory

Description:

A description of the error that caused a procedure call to be terminated. This symbol is used with a procedure_terminated, when the system exceeded the amount of memory specified in the option_max_memory option given in the corresponding procedure call. It carries one argument: An OMSTR, which may be empty.

Example:

xml prefix mathml

procedure_terminated (error_memory)

Signatures:

sts

[Next: error_runtime] [Previous: info_message] [Top]

---

error_runtime

Description:

A description of the error that caused a procedure call to be terminated. This symbol is used with a procedure_terminated, when the system exceeded the runtime specified in the option_runtime option given in the corresponding procedure call. It carries one argument: An OMSTR, which may be empty. Note that this symbol is not intended to be used when a different runtime error occurred. In those cases, one should use error_system_specific.

Example:

xml prefix mathml

procedure_terminated (error_runtime)

Signatures:

sts

[Next: error_system_specific] [Previous: error_memory] [Top]

---

error_system_specific

Description:

A description of the error that caused a procedure call to be terminated. This symbol is used with a procedure_terminated, when the error is specific to the system that carried out the calculation. This error

must carry exactly one argument, and it must be an OMSTR describing the error that occurred.

Example:

xml | prefix | mathml

procedure_terminated (error_system_specific Error, the group identification for groups of size\n 3628800 is not available called from\n <function>(<arguments>) called from read-eval-loop\n Entering break read-eval-print loop ...\n you can 'quit;' to quit to outer loop, or\n you can 'return;' to continue\n brk>\n)

Signatures:

sts

[First: procedure_call] [Previous: error_runtime] [Top]

Home Overview Documents Content Dictionaries Software & Tools The OpenMath Society OpenMath Projects OpenMath Discussion Lists OpenMath Meetings Links

---

OpenMath Content Dictionary: scscp2

Canonical URL:
http://www.win.tue.nl/SCIEnce/cds/scscp2.ocd
CD File:
scscp2.ocd
CD as XML Encoded OpenMath:
scscp2.omcd
Defines:
get_allowed_heads, get_service_description, get_signature, get_transient_cd, is_allowed_head, no_such_transient_cd, retrieve, service_description, signature, store_persistent, store_session, symbol_set, symbol_set_all, unbind
Date:
2009-06-25
Version:
1 (Revision 7)
Review Date:
Status:
experimental

This CD defines symbols for the description of the management of mathematical queries. In particular, it is used by the SCIEnce project [SCIEnce website] in the communication between a web service (i.e. computer algebra system, proof checker, etc) and a client. SCSCP is an abbreviation for "Symbolic Computation Software Composability Protocol" [SCSCP specification].

The objects in this CD are somewhat more sophisticated than those in scscp1, and some SCSCP compliant applications may not support these. In particular, we add support for so-called "transient CDs", allowing a server to refer to symbols from temporary content dictionaries, valid only for the duration of the session. Please refer to the specification for more information on this concept.

The symbols in this CD mainly serve two purposes: working with remote objects (scscp2.store_session , scscp2.store_persistent , scscp2.retrieve, scscp2.unbind) and determining the procedures a system supports (scscp2.get_allowed_heads, scscp2.is_allowed_head, scscp2.get_transient_cd, scscp2.get_signature, scscp2.get_service_description, scscp2.signature, scscp2.service_description). There are also some special symbols (scscp2.symbol_set, scscp2.symbol_set_all, scscp2.no_such_transient_cd)

This version of the Content Dictionary agrees with version 1.3 of the SCSCP protocol.

---

store_session

Description:
This indicates the request to store an object on the server side (possibly after computing or simplifying it), returning only a cookie (actually, OM reference) pointing to an object that is usable (using an OMR) in the remainder of the current SCSCP session to get access to the actual object.

The client could ask:

Example:

xml prefix mathml

procedure_call (store_session (6177887))

The server might then reply:

Example:

xml prefix mathml

procedure_completed ( )

Note that the content of the OMR may vary, e.g. the URI does not necessarily start with scscp://.

Signatures:

sts

[Next: store_persistent] [Last: no_such_transient_cd] [Top]

---

store_persistent

Description:

This indicates the request to store an object on the server side (possibly after computing or simplifying it), returning only a cookie (actually, OM reference) pointing to an object that is usable (using OMR) in the foreseeable future, possibly from different sessions, to get access to the actual object. The server is encouraged to describe the expected lifetime of this object and whether references to this object from different SCSCP sessions are allowed in the response to a scscp2.get_signature request on this symbol. However, at this time we provide no automated or machine-readable mechanism for handling these lifetimes.

Signatures:

sts

[Next: retrieve] [Previous: store_session] [Top]

---

retrieve

Description:

Using the cookie that was obtained earlier by calling the scscp2.store_session or scscp2.store_persistent procedure or another procedure call, return to the client an OM object representing the object, referred by the cookie.

The client could ask:

Example:

xml prefix mathml

procedure_call (retrieve ( ))

The server might then reply:

Example:

xml prefix mathml

procedure_completed (6177887)

Signatures:
sts

---

unbind

Description:
This indicates the request to remove the object, referred by the cookie, from the server.

Example:
xml prefix mathml

procedure_call (unbind ( ))

The server might then reply:

Example:
xml prefix mathml

procedure_completed ( )

Signatures:
sts

[Next: get_allowed_heads] [Previous: retrieve] [Top]

get_allowed_heads

Description:
This symbol is used to find the list of procedures supported by an SCSCP server.

The client could send:

Example:
xml prefix mathml

procedure_call (get_allowed_heads ( ))

and the server might then reply:

Example:
xml prefix mathml

procedure_completed (symbol_set (GroupIdentificationService , group , CDName (permut1) , CDGroupName (scscp)))

indicating that it accepts the symbol GroupIdentificationService from the transient CD scscp_transient_1, the symbol group1.group, the entire permut1 CD, and all cds from the CD group called scscp.

Signatures:
sts

[Next: is_allowed_head] [Previous: unbind] [Top]

---

is_allowed_head

Description:

This symbol is used to find whether a particular procedure is supported by an SCSCP server. The reply must be either true or false, described in one of the appropriate symbols from the logic1 content dictionary.

The client could ask:

Example:

xml prefix mathml

procedure_call (is_allowed_head (+))

and the server might then reply:

Example:

xml prefix mathml

procedure_completed (F)

indicating that it does not accept this symbol. Another, slightly more contrived, example would be for the client to ask:

Example:

xml prefix mathml

procedure_call (is_allowed_head (is_allowed_head))

and the server to reply:

Example:

xml prefix mathml

procedure_completed (T)

In particular, this is the method of choice to find out whether a particular server supports storing remote objects using the scscp2.store_session and/or scscp2.store_persistent methods.

Signatures:

sts

[Next: get_transient_cd] [Previous: get_allowed_heads] [Top]

---

get_transient_cd

Description:

This symbol is used to get the contents of a transient CD created by a server.

The client could send:

Example:

xml prefix mathml

procedure_call (get_transient_cd (CDName (scscp_transient_1)))

and the server might then reply:

Example:

xml | prefix | mathml

```
procedure_completed ( CD ( CDName ( scscp_transient_1 ) , CDDate ( 2007-08-24 ) , Description (
CD created by the service provider ) , CDDefinition ( Name ( GroupIdentificationService ) , Description
( IdGroup(permgroup by gens) ) ) ) )
```

Signatures:

sts

[Next: get_signature] [Previous: is_allowed_head] [Top]

---

get_signature

Description:

A symbol for the client to inquire about the signature of a particular function.

The client could send:

Example:

xml | prefix | mathml

```
procedure_call ( get_signature ( GroupIdentificationService ) )
```

and the server might then reply with a signature message.

Signatures:

sts

[Next: get_service_description] [Previous: get_transient_cd] [Top]

---

get_service_description

Description:

A symbol for the client to ask for some description of a service. Note that this is a very generic description of the service running on a particular port on a particular machine. More details about for example the available symbols there may be obtained with get_allowed_heads, get_signature or get_transient_cd.

The client could send:

Example:

xml | prefix | mathml

```
procedure_call ( get_service_description ( ) )
```

Signatures:

sts

[Next: signature] [Previous: get_signature] [Top]

---

signature

Description:
The symbol to use for describing the types of arguments of a particular function.

Example:

xml prefix mathml

procedure_completed (signature (GroupIdentificationService , 1 , 1 , symbol_set (group , CDName (permut1))))

This means that this GroupIdentificationService requires at least 1 argument, and at most 1 argument, and that the symbol group1.group or anything from the permut1 CD may be used to form this argument.

Example:

xml prefix mathml

procedure_completed (signature (CAS_Service , 0 , ∞ , (CDGroupName (scscp) , CDName (scscp_transient_0) , CDName (scscp_transient_1) , CDName (arith1) , CDName (transc1))))

indicating that this particular CAS_Service takes any number of arguments, which may be formed using anything from the CD group scscp, one of two transient CDs, and the arith1 or transc1 CD.

Signatures:
sts

[Next: service_description] [Previous: get_service_description] [Top]

---

service_description

Description:
The symbol for the server to use in a response to scscp2.get_service_description. It takes three OMSTR arguments: Name, Version, and Description.

Example:

xml prefix mathml

procedure_completed (service_description (MyGreatService , 1.1.0 , This service does fantastic things!))

Signatures:
sts

[Next: symbol_set] [Previous: signature] [Top]

---

symbol_set

Description:
This symbol is used in the reply to a scscp2.get_allowed_heads call. It should be the head of an OM Application, the contents of the OMA being arbitrarily many OM Symbols (meaning that a particular symbol is supported), OMA's with head meta.CDName (meaning that all symbols of a particular CD are supported) or OMA's with head meta.CDGroupName (meaning that all symbols of all CDs of a particular CD group are supported). See the example at scscp2.get_allowed_heads.

Signatures:
sts

[Next: symbol_set_all] [Previous: service_description] [Top]

symbol_set_all

Description:
This symbol is used in the reply to a scscp2.get_signature message. It means that this particular service takes any OpenMath object as argument.

A reply might be:

Example:
xml | prefix | mathml

procedure_completed (signature (Something , 0 , ∞ , symbol_set_all))

indicating that this service, scscp_transient_1.Something, takes between 0 and infinity arguments, each of which can be of any type.

Signatures:
sts

[Next: no_such_transient_cd] [Previous: symbol_set] [Top]

no_such_transient_cd

Description:
Used for errors that arise when the client asks for a transient cd that the server cannot handle.

Example:
xml | prefix | mathml

no_such_transient_cd scscp_transient_7

Signatures:
sts

[First: store_session] [Previous: symbol_set_all] [Top]

The MathML CD Group: Proposed update for MathML3

David Carlisle

NAG, W3C Math WG
davidc@nag.co.uk

Abstract. The MathML3 draft (an editors' copy may be found at `http://monet.nag.co.uk/~dpc/draft-spec` formalises the relationship between Content MathML elements and OpenMath core CDs. While editing this specification a small number of additions to the MathML CD Group have been proposed, either to clarify existing elements or to model new features in MathML.
The proposed new CDs are visible at `http://monet.nag.co.uk/~dpc/cdfiles2`.

1 Introduction

A small number of additional symbols, and one additional Content Dictionary are proposed to be added to the MathML CD Group.

2 calculus1

The **partialdiff** symbol in the **calculus1** CD and the partialdiff element in MathML are structurally very different. This has always made mapping between the two very difficult.

In MathML, the variables are given by name (and optonally a degree), but for **partialdiff**, the variables are specified by position, with degree being indicated by repetition.

So in

$$\frac{\partial^3}{\partial x^2 \partial x}$$

The variables are prepresnted for **partialdiff** by the list (1,1,3) (x twice and z). This is managable for explict integer degrees, but becomes increasingly hard to come up with a representation for degress given symbolically.

To represent

$$\frac{\partial^n}{\partial x^j \partial x^k}$$

using **partialdiff** you need to use a list constructor to generate a list of j 1's and k 3's and have no natural represntation for the total degree being n (as the total degree is just given by the length of the list of variable slots, $j + k$ here.

We propose to add a **partialdiffdegree** sybnol to **calculus1** which closely models the MathML usage, so this last case would be specified by a list of degrees (m, n) and a total degree n.

3 mathmlattr

MathML "XML" Attrubutes, in addition to the sematic annotations of the of the semantics element (which correspond naturally to OpenMath attributions. These cause some problems in converting between the formats. While `<cn type='integer'>1</cn>` corresponds naturally to `<OMI>1</OMI>` it is not so imediatey clear what should be the OpenMath version of
`<cn type='integer' class='abc' xlink:href='http://example.com'>1</cn>`.

We propose to add a new Content Dictionary with symbols to be used as attribution keys. One symbol will be added for each MathML attribute (class, style, other, definitionURL) plus an additional symbol to encode any "foreign" namespace attribute. thus the cn example would be encoded as

```
<OMATTR>
  <OMATP>
   <OMS cd="mathmlattr" name="class"/>
```

```
    <OMSTR>abc</OMSTR>
    <OMS cd="mathmlattr" name="foreign"/>
      <OMA><OMS cd="mathmlattr" name="foreign_attribute"/>
     <apply><csymbol cd="mathmlattributes">foreign_attribute</csymbol>
       <OMSTR>http://www.w3.org/1999/xlink</OMSTR>
       <OMSTR>xlink</OMSTR>
       <OMSTR>href</OMSTR>
       <OMSTR>http://example.com</OMSTR>
     </apply>
   </annotation-xml>
   <OMI>1</OMI>
</OMATTR>
```

where the class attribute is encoded as an attribution with the **class** symbol, and the xlink:href attribute is an attribution with the **foreign** symbol with a term construucted with **foreign_attribute** which encodes, the namespace, prefix, local name and value of the attribute.

4 fns1

The **fns1** CD includes a symbol **domainofapplication** that is, at best, incompletely defined. It is defined as being used for compatibatility with the MathML domainofapplication element, however the STS type signature given is identical to thet of **domain**. So this appears to be synonymous with the **domain** symbol, which takes a function and *returns* its domain. The domainofapplication elemnt is used (mainly) in MathML to *restrict* a function to a specified domain, so it's signature is dual to that of **domain**, taking a function and a domain and returning a function. This restriction operation is genarally useful, not just for MathML compatibility but does not appear in the current core CDs. we propose to add **restriction** to **fns1** and deprecate (but keep) the existing **domain** symbol.

5 fns2

Some of the examples make use of a symbol **make_list** from the **list1** CD but there is no such symbol. We propose to just fix this as a minor editorial error.

We propose to add the symbol **predicate_on_list** which takes arguments a binary predicate and a seequence and returns the conjunction of applying the predicate to all pairs (in order). this allows a direct encoding of *chains* of binary predicates $a < b < c < d$.

6 nums1

The **nums1** CD includea a constructor for based integers, taking an OMString of digits and an integer base. We propose to add **based_float** which is the same

(except that the string is allowed to include "."). This is probably not going to be widely used, but greatly simplifies the conversion to and from MathML, which allows `<cn type='real' base='16'>A.8</cn>` as a representation of 15.5_{10}.

7 interval1

Both MathML and OpenMath have previously suggested that the limits on a definite interval should be understood to be representing the end points of an interval. This usage requires that the interval has an ordering or path structure, so that integrating from a to b is different from integrating from b to a. We propose to add a **ordered_interval** element to the **interval1** CD. (The detailed encoding has been developed by James Davenport and described elesewhere in these proceedings, but this addition is listed here for completeness.

OpenMath Content Dictionaries for SI Quantities and Units

Joseph B. Collins

Naval Research Laboratory
4555 Overlook Ave, SW
Washington, DC 20375-5337
joseph.collins@nrl.navy.mil

Abstract. We document the creation of a new set of OpenMath content dictionaries to support the expression of quantities and units under the International System of Units (SI). While preserving many of the concepts embodied in the original content dictionaries, these new content dictionaries provide a foundation for quantities and units that is compliant with international standards. We respond to questions raised in prior efforts to create content dictionaries for units and dimensions by proposing and applying some rationalized criteria for the creation of content dictionaries in general. The results have been released and submitted to the OpenMath website as contributed content dictionaries.

Content Dictionaries for Algebraic Topology*

Jónathan Heras, Vico Pascual, and Julio Rubio

Departamento de Matemáticas y Computación, Universidad de La Rioja,
Edificio Vives, Luis de Ulloa s/n, E-26004 Logroño (La Rioja, Spain).
{jonathan.heras, vico.pascual, julio.rubio}@unirioja.es

Abstract. Kenzo is a Symbolic Computation System devoted to Algebraic Topology that works with the main mathematical structures used in this discipline. In this paper, we present OpenMath Content Dictionaries for each mathematical structure in Algebraic Topology the Kenzo system works with. Besides, how using them to interoperate with a particular Theorem Prover and to obtain certified calculations from Kenzo is explained.

1 Introduction

Kenzo [2] is a Common Lisp system devoted to Symbolic Computation in Algebraic Topology. It was developed in 1997 under the direction of F. Sergeraert and has been successful, in the sense that it has been capable of computing homology groups unreachable by any other means.

Up to now, the mathematical structures Kenzo works with have not been represented in OpenMath [1]. In order to communicate Kenzo with other systems, like GAP [3] or ACL2 [6], we have tackled the goal of developing these OpenMath Content Dictionaries, previous works in these directions are [7] and [4].

2 Description of Kenzo Content Dictionaries

The Kenzo system works with the main mathematical structures used in Simplicial Algebraic Topology, [5]. Figure 1 shows the Kenzo class diagram where each class corresponds to the respective mathematical structure. For each one of these mathematical structures, an OpenMath Content Dictionary has been defined (these Content Dictionaries are available at `http://www.unirioja.es/cu/joheras/cdgroup.html`).

Now, we are going to focus on the simplicial sets Content Dictionary; the rest are based on the same ideas.

* Partially supported by Universidad de La Rioja, project API08/08, and Ministerio de Educación y Ciencia, project MTM2006-06513.

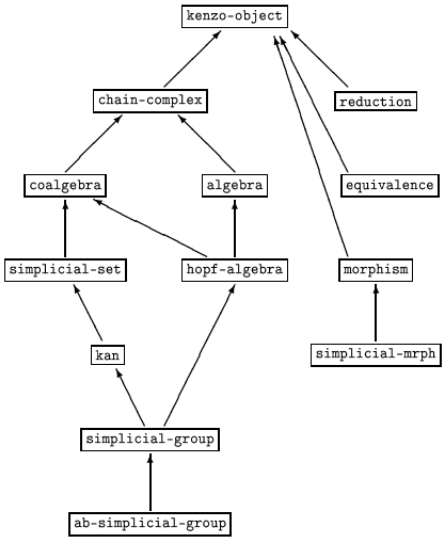

Fig. 1. Class diagram of the Kenzo system.

Definition 1. A *simplicial set* K, [5], is a disjoint union $K = \bigcup_{q\geq 0} K^q$, *where the* K^q are sets, together with functions

$$\partial_i^q : K^q \to K^{q-1},\ q > 0,\ i = 0, \ldots, q,$$
$$\eta_i^q : K^q \to K^{q+1},\ q \geq 0,\ i = 0, \ldots, q,$$

subject to relations

$$\begin{array}{rcll}
\partial_i^{q-1}\partial_j^q &=& \partial_{j-1}^{q-1}\partial_i^q, & i < j \\
\eta_i^{q+1}\eta_j^q &=& \eta_j^{q+1}\eta_{i-1}^q, & i > j \\
\partial_i^{q+1}\eta_j^q &=& \eta_{j-1}^{q-1}\partial_i^q, & i < j \\
\partial_i^{q+1}\eta_i^q &=& \partial_{i+1}^{q+1}\eta_i^q, & \textit{identity} \\
\partial_i^{q+1}\eta_j^q &=& \eta_j^{q-1}\partial_{i-1}^q, & i > j+1
\end{array}$$

The functions ∂ and η are called the *face operators* and the *degeneracy operators* respectively.

To define a simplicial set, we must provide the disjoint sets $\{K^q\}_{q\geq 0}$ and the face and degeneracy operators. The sets $\{K^q\}_{q\geq 0}$ can be seen as a graded set so it is possible to consider its characteristic function which, from an element x and a degree g, determines if the element x belongs to the set K^g. To be precise, an *invariant* function can be used in order to encode the characteristic function of the graded set $\{K^q\}_{q\geq 0}$.

Based on the previous way of representation, the following signature has been defined for simplicial sets.

```
inv  : u   nat       -> bool
```

```
face : u   nat   nat -> u
deg  : u   nat   nat -> u
```

where u denotes the Universe, of Lisp objects in this case.

The OpenMath signature of Figure 2 corresponds with the previous one:

```
<Signature name="simplicial-set">
    <OMOBJ xmlns="http://www.openmath.org/OpenMath">
        <OMA>
            <OMS name="mapsto" cd="sts"/>
            <OMA id="inv">
                <OMS cd="sts" name="mapsto"/>
                <OMV name="Simplicial-Set-Element"/>
                <OMV name="PositiveInteger"/>
                <OMS cd="setname2" name="boolean"/>
            </OMA>
            <OMA id="face">
                <OMS cd="sts" name="mapsto"/>
                <OMV name="Simplicial-Set-Element"/>
                <OMV name="PositiveInteger"/>
                <OMV name="PositiveInteger"/>
                <OMV name="Simplicial-Set-Element"/>
            </OMA>
            <OMA id="degeneracy">
                <OMS cd="sts" name="mapsto"/>
                <OMV name="Simplicial-Set-Element"/>
                <OMV name="PositiveInteger"/>
                <OMV name="PositiveInteger"/>
                <OMV name="Simplicial-Set-Element"/>
            </OMA>
            <OMV name="Simplicial-Set"/>
        </OMA>
    </OMOBJ>
</Signature>
```

Fig. 2. Signature of simplicial-set.

The formal mathematical properties of the simplicial sets are given in the <FMP> tag of the simplicial-set definition. Note that, besides the simplicial properties, some invariance properties of face and degeneracy operators and the relations among them must be added. All of them have also been included in natural language by using the <CMP> tags. For instance, the face operator invariance has been represented as in Figure 3.

Finally, an example of simplicial set has been included. Namely, the simplicial set with only one element, nil, belonging to each set K^q and with each face operation of degree q returning the element of degree $q-1$ has been considered, a piece of this example can be seen in Figure 4.

```
<CMP> The face operator is invariant </CMP>
<FMP>
...
<OMA>
    <OMS cd="logic1" name="implies"/>
    <OMA>
        <OMV name="inv"/>
        <OMV name="x"/>
        <OMV name="q"/>
    </OMA>
    <OMA>
        <OMV name="inv"/>
        <OMA>
            <OMV name="face"/>
            <OMV name="x"/>
            <OMV name="i"/>
            <OMV name="q"/>
        </OMA>
        <OMA>
            <OMS cd="arith1" name="minus"/>
            <OMV name="q"/>
            <OMI>1</OMI>
        </OMA>
    </OMA>
</OMA>
...
</FMP>
```

Fig. 3. A Formal Mathematical Property of Simplicial Sets.

```
<example>
...
    <OMBIND>
        <OMS name="face"/>
        <OMBVAR>
            <OMV name="x"/>
            <OMV name="i"/>
            <OMV name="q"/>
        </OMBVAR>
        <OMS cd="list" name="nil"/>
    </OMBIND>
...
</example>
```

Fig. 4. Fragment of the example for Simplicial Sets.

More complicated examples than the previous one can be included without difficulties. The reason to provide such a simple example, like the previous one, is that we have used our OpenMath Content Dictionaries to integrate Kenzo with a

Theorem Prover, namely with ACL2 [6] in order to obtain certified computations from Kenzo.

The ACL2 tool used for dealing with axiomatic structures is that of *encapsulate*. An encapsulate has a list of function signatures and some properties of the encapsulated functions. In addition, ACL2 demands giving a *witness* for the set of functions, that is, an instance satisfying the properties. This ensures that the encapsulate has at least one *model* which avoids the appearance of inconsistencies in the ACL2 logic.

Obviously, there exists a relationship between each Content Dictionary and the respective encapsulate for the same mathematical structure. In this line, an interpreter to extract the encapsulate from each Content Dictionary has been developed. Now, a fragment of an encapsulate for Simplicial Sets is shown in Figure 5.

```
(encapsulate

 (((inv * *) => *)
  ((face * * *) => *)
  ((degeneracy * * *) => *)
  )

 (local (defun inv (x q)
          (declare (IGNORE q))
          (equal x nil)))

 (local (defun face (x i q)
          (declare (IGNORE x i q))
          nil))

 (local (defun deg (x i q)
          (declare (IGNORE x i q))
          nil))

 ; ... lines skipped

 (defthm prop5
     (implies (and (inv x q) (< i j))
              (equal (face (deg x j q) i (+ q 1))
                     (deg (face x i q) (- j 1) (- q 1)))))

 )
```

Fig. 5. Encapsulate of Simplicial Sets.

In this way, besides representing some of the main concepts used in Algebraic Topology, our Content Dictionaries allow us to interact with the ACL2 Theorem Prover.

Moreover, some mathematical structures (such as spheres, Moore spaces, loop spaces and so on) are predefined objects in the Kenzo system. These objects have been included in the corresponding Content Dictionary in a descriptive way. For instance, spheres are Simplicial Sets, and its Kenzo representation is given by means of a function with a natural number as argument (constructing the corresponding Simplicial Set). We show an example in Figure 6.

```
<CDDefinition>
        <Name>sphere</Name>
        <Description>
            This symbol is a function with one argument, which should
            be a natural number n between 1 and 14. When applied to
            n it represents the sphere of dimension n.
        </Description>
        ...
        <FMP>
            <OMOBJ xmlns="http://www.openmath.org/OpenMath">
                <OMA>
                    <OMS cd="logic1" name="implies"/>
                    <OMA>
                        <OMS cd="Simplicial-Sets" name="sphere"/>
                        <OMV name="n"/>
                    </OMA>
                    <OMA>
                        <OMS cd="logic1" name="and"/>
                        <OMA>
                            <OMS cd="set1" name="in"/>
                            <OMV name="n"/>
                            <OMS name="N" cd="setname1"/>
                        </OMA>
                        <OMA>
                            <OMS cd="relation1" name="leq"/>
                            <OMI>1</OMI>
                            <OMV name="n"/>
                        </OMA>
                        <OMA>
                            <OMS cd="relation1" name="leq"/>
                            <OMV name="n"/>
                            <OMI>14</OMI>
                        </OMA>
                    </OMA>
                </OMA>
            </OMOBJ>
        </FMP>
</CDDefinition>
```

Fig. 6. Representation of spheres.

By using the encapsulate obtained from the simplicial sets Content Dictionary it is possible to prove that each Kenzo sphere is really a Simplicial Set.

3 Conclusions

In this paper, we have presented some OpenMath Content Dictionaries where the main mathematical structures used in Simplicial Algebraic Topology have been defined. The definitions given in these Content Dictionaries include the axiomatic parts and have been used, for example, to interoperate with deduction systems. In this way, a gate has been opened not only to communicate with other systems which work with the same mathematical structures but also to prove the correctness of some constructions or calculations carried out by the Kenzo system. For instance, when Kenzo builds an object belonging to the simplicial-set class, that it is really a simplicial set can be proved. In this way, some calculations with certificates can be carried out.

References

1. Buswell S., Caprotti O., Carlisle D.P., Dewar M.C., Gaëtano M., Kohlhase M. *OpenMath* Version 2.0, 2004. `http://www.openmath.org/`.
2. Dousson X., Sergeraert F., Siret Y., *The Kenzo program*, Institut Fourier, Grenoble, 1999. `http://www-fourier.ujf-grenoble.fr/~sergerar/Kenzo/`.
3. GAP - Groups, Algorithms, Programming - a System for Computational Discrete Algebra. `http://www.gap-system.org/`.
4. Heras J., Pascual V., Rubio J., *Using Open Mathematical Documents to interface Computer Algebra and Proof Assistant systems*. To appear in Proceedings of MKM 2009.
5. Hilton P. J., Wylie S., *Homology Theory*, Cambridge University Press, (1967).
6. Kaufmann M., Manolios P., Moore J., *Computer-Aided Reasoning: An Approach. Kluwer Academic Press*, Boston (2000).
7. Romero A., Ellis G., Rubio J., *Interoperating between Computer Algebra systems: computing homology of groups with Kenzo and GAP*. To appear in Proceedings of ISSAC 2009.

Quantifiers and Big Operators in OpenMath

James H. Davenport[1] and Michael Kohlhase[2]

[1] Department of Computer Science
University of Bath, Bath BA2 7AY, United Kingdom
J.H.Davenport@bath.ac.uk
[2] School of Engineering & Science, Jacobs University Bremen
Campus Ring 12, D-28759 Bremen, Germany
m.kohlhase@jacobs-university.de

Abstract. The effort to align MathML 3 and OpenMath has led to a realisation that (pragmatic) MathML's `condition` and `domainofapplication` elements, when used with quantifiers, do not have a *neat* expression in OpenMath.
This paper analyzes the situation focusing on quantifiers and proposes a solution, via six new symbols. Two of them fit completely within the existing OpenMath structure, and we place them in the associated `quant3` CD. The others require a generalization of `OMBIND`.
We also propose, logically separately but in the same area, a `quant2` CD with `existsuniquely`, commonly written ∃!, and the 'fusion' symbol `existsuniquelyin`.
In a second step we generalize the solution to the phenomenon of big operators that MathML 2 implicitly provides but which do not have a direct counterpart in the OpenMath CDs.

1 Introduction

The effort to align MathML 3 [Con08] and OpenMath [Con04] has led to a realisation that (pragmatic) MathML 3's `condition` and `domainofapplication` elements, when used with quantifiers, does not have a *neat* expression in OpenMath. We have described the MathML 3/OpenMath 3 and the general representational issues in [DK09], which we will assume as background reference. But a central part of the alignment effort remains unsolved there: the provisioning of the OpenMath content dictionaries that supply the necessary symbols. We will describe what needs to be done on this front in this paper, which should be seen as a companion paper to [DK09].

We start out by analyzing the situation focusing on quantifiers and then generalizing the solution to the phenomenon of big operators that MathML 2 implicitly provides but which do not have a direct counterpart in the OpenMath CDs.

2 Existential and Universal Quantifiers

"$\forall n \in \mathbb{N} \ldots$" is a very natural piece of mathematical notation, even though it tends not to be formally defined. MathML (Content) can represent this via the

`condition` element, but OpenMath[3] has hitherto had no *direct* way of doing so, relying on the following equivalences:

$$\forall v \in S.p(v) \Leftrightarrow \forall v.(v \in S) \Rightarrow p(v) \quad (1)$$

$$\exists v \in S.p(v) \Leftrightarrow \exists v.(v \in S) \wedge p(v) \quad (2)$$

However, in practice[4], it would be better to have a convenient shorthand for these, hence this proposal.

The challenge is the syntax of OpenMath. One represents $\forall n.p(n)$ as

```
1 <OMBIND>
    <OMS name="forall" cd="quant1"/>
    <OMBVAR><OMV name="n"/></OMBVAR>
    <OMA><OMV name="p"/><OMV name="n"/></OMA>
  </OMBIND>
```

Where is the "$\in \mathbb{N}$" going to fit in this syntax? In [DK09] we have argued that there are actually two representation styles that need to be supported: Figure 1 gives both styles the *first-order* style on the left makes use of expressions with bound variables whereas the *higher-order* style directly uses sets. The latter is

```
<OMBIND>                                  <OMBIND>
  <OMS name="forallcond" cd="quant3"/>      <OMA>
  <OMBVAR>                                    <OMS name="forallin" cd="quant3"/>
    <OMV name="n"/>                           <OMS name="N" cd="setname1"/>
  </OMBVAR>                                 </OMA>
  <OMA>                                     <OMBVAR>
    <OMS name="in" cd="set1"/>                <OMV name="n"/>
    <OMV name="n"/>                         </OMBVAR>
    <OMS name="N" cd="setname1"/>           p(n)
  </OMA>                                  </OMBIND>
  p(n)
</OMBIND>
```

here and in the following we use boxed mathematical notation to represent the obvious OpenMath counterparts

Fig. 1. Two representations of quantifiers with domains

nearer to our example, and we propose to use the fact that the first child of an `OMBIND` need not be an atom and propose a symbol `forallin` which acts as a binding constructor, i.e. an operator that takes a set as an argument and returns a binding operator, which can then be used in the `OMBIND`. The first-order style of representation could be glossed in Mathematics as $\forall_c n : n \in \mathbb{N}.p(n)$; it makes the bound variable that was implicit in the higher-order construction explicit in a condition expression. So we would contend that the higher-order style is closer to our original example. But the first-order style is more in other situations, e.g. $\forall x : (1/x \text{ is irrational}).p(x)$, which can be represented as $\boxed{\forall x : (\forall n, d.n/d \neq x).p(x)}$.

[3] If restricted only to the content dictionaries the authors know about.

[4] A referee objected to [DK09] on the grounds that it had stated that such shorthand was not necessary, writing 'one might [as well] write "not p" as"p implies false"'.

Listing 1.1. A natural case for a quantifier with a condition

```
<OMBIND>
  <OMS name="forallcond" cd="quant3"/>
  <OMBVAR><OMV name="x"/></OMBVAR>
  <OMBIND>
    <OMS name="forall" cd="quant1"/>
    <OMBVAR><OMV name="n"/><OMV name="d"/></OMBVAR>
    <OMA><OMS cd="relation1" name="ne"/>
      <OMA><OMS cd="arith1" name="divide">
        <OMV name="n"/>
        <OMV name="d"/>
      </OMA>
      <OMV name="x"/>
    </OMA>
  </OMBIND>
  p(x)
</OMBIND>
```

Note that for the first-order representations on the left of Figure 1 and in Listing 1.1 we need the extension of binding objects to allow multiple scopes proposed in [DK09]. We consider the examples in this paper and representation possibilities they enable to be a good reason for this extension.

The symbols `forallin` and `forallcond` are defined in the proposed `quant3` CD, with Formal Mathematical Properties corresponding to equations (1) and (2), at least in the single-variable case. They are related by the following FMPs:

$$\begin{aligned}\forall P, Q.\ [\forall\!\!\!\!_{\,c}\, x.Q(x)P(x)] &\Leftrightarrow [\forall\!\!\!\!_{\,i}(\lambda z.P(z))]x.Q(x)\\ \forall Q, S.\ [\forall\!\!\!\!_{\,i}(S)]x.Q(x) &\Leftrightarrow \forall x.(x \in S) \Rightarrow Q(x)\end{aligned}$$

where we use $\forall\!\!\!\!_{\,c}$ for `forallcond` and $\forall\!\!\!\!_{\,i}$ for `forallin`. We should note what `forallin` does, and does not, encode.

does $\forall n \in \mathbb{N}$, $\forall m, n \in \mathbb{N}$, $\forall x \in [0,1]$, $\forall x \in (0,\infty)$ (but not the equivalent $\forall x > 0$).

does not $\forall n \in \mathbb{N}, x \in \mathbb{R}$ (this needs two nested bindings of `forallin`[5]), $\forall n > 2$ (though we *can* encode $\forall n \in (2,\infty)$), $\forall m < n \in \mathbb{N}$ (though we *can* encode $\forall n \in (-\infty, n)$ and use `integer_interval`).

[Con03, section 4.2.5.1] gives an example for the formula

$$\forall x \in \mathbb{N}.\exists p, q \in \mathbb{P}.p + q = 2x$$

where $\mathbb{P}$ stands for the set of prime numbers. While this would succumb to forall/implies and exists/and encodings, it is a better tribute to the power of our extended quantifiers to use the following encoding:

```
<OMBIND>
  <OMA>
    <OMS name="forallin" cd="quant3"/>
    <OMS name="N" cd="setname1"/>
  </OMA>
```

[5] At first sight it seems that this could be represented as $\forall\!\!\!\!_{\,i}\mathbb{N} \times \mathbb{R}$. but we contend that these are different. We can encode $\forall z \in (\mathbb{N} \times \mathbb{R})$, and *later* destructure z, but OpenMath doesn't have a destructuring bind.

```
  <OMBVAR> <OMV name="x"/> </OMBVAR>
  <OMBIND>
    <OMA>
      <OMS name="existsin" cd="quant3"/>
      <OMS name="P" cd="setname1"/>
   </OMA>
    <OMBVAR><OMV name="p"/><OMV name="q"/></OMBVAR>
    p + q = 2x
  </OMBIND>
</OMBIND>
```

Note that in some cases, we naturally combine these two: We often see something like the following $\forall x, y \in \mathbb{R} : x - y \neq 0.\frac{1}{x-y} \in \mathbb{R}$. This would be suitably encoded as

```
<OMBIND>
  <OMA>
    <OMS name="forallincond" cd="quant3"/>
    ℝ
  </OMA>
  <OMBVAR><OMV name="x"/><OMV name="y"/></OMBVAR>
  1/(x−y) ∈ ℝ
  x − y ≠ 0
</OMBIND>
```

using the a symbol `forallincond` that we propose to add to `quant3` content dictionary together with the `FMP`

$$\forall_{ic}(S)x : C(x).D \Leftrightarrow \forall_i(S).C(x) \Rightarrow D$$

The existential variants `existin`, `existscond`, and `existsincond` of all three have analogous FMPs in the `quant3` content dictionary.

3 Unique Existence

Although the notation $\exists!$ is relatively new[6], it is convenient. It would be easy enough to introduce into OpenMath, as a symbol, say `existsuniquely` in the `quant2` CD, with one property, which could be held to define it.

$$\exists!x.p(x) \Leftrightarrow (\exists x.p(x)) \wedge ((p(x) \wedge p(y)) \Rightarrow x = y)\,. \tag{3}$$

One might naturally ask: "what about $\exists!x \in \mathbb{N} : \ldots$, and similar constructs". There is obviously no difficulty (other than the length of the name!) in adding `existsuniquelyin` to the `quant3` CD. The real challenge is what semantics does one want. There are two options.

"It's unique *and* **it's in** $\mathbb{N}$**"**

$$\exists!x \in \mathbb{N}.p(x) \Leftrightarrow (\exists x.(x \in \mathbb{N} \wedge p(x)) \wedge ((p(x) \wedge p(y)) \Rightarrow x = y)\,. \tag{4}$$

This is what one would get by applying equation (2) to the $\exists$ on the right-hand side of equation (3).

[6] It was not in use when the first author was a student, and the earliest we can trace it to is [Bar84, p. xiii].

"It's unique *within* $\mathbb{N}$**"**

$$\exists! x \in \mathbb{N}.p(x) \Leftrightarrow (\exists x.(x \in \mathbb{N} \wedge p(x))) \wedge (((p(x) \wedge x \in \mathbb{N}) \wedge (p(y) \wedge y \in \mathbb{N})) \Rightarrow x = y)\,. \tag{5}$$

This is what one would get by applying equation (3) to equation (2).

While both have their part to play, it appears that (5) is the one more commonly met, and we propose to add this meaning to `quant3`.

4 Other Similar cases: Big Operators

Note that in the analysis above, the fact that we are dealing with quantifiers is secondary, the interesting part is the fact that, the "operators take qualifiers" `condition` and `domainofapplication` in MathML 2. The full list of these operators can be grouped into three parts: the special operators `int`, `sum`, `product`, `root`, `diff`, `partialdiff`, `limit`, `log`, `moment`, `forall`, `exists`, the binary/$n$ary operators `plus`, `times`, `max`, `min`, `gcd`, `lcm`, `mean`, `sdev`, `variance`, `median`, `mode`, `and`, `or`, `xor`, `union`, `intersect`, `cartesianproduct`, `compose`, as well as the relational operators `eq`, `leq`, `lt`, `geq`, `gt`. The use of qualifiers with relational operators is being deprecated in MathML 3, so we will not concern ourselves with these.

The reason binary operators can take qualifiers is that they can be "lifted" to their respective "big operator form", for instance $\bigcup$ for $\cup$. Note that if we look at the special operators in the first group, then we can see that `sum`, `product`, `forall`, and `exists` are the conventionalized "big operators" for `plus`, `times`, `and`, and `or`. We have covered the quantifiers above, so let us look whether `sum` and `product` show the same behavior. If they do, the quantifiers may give a good model for the other "big operators". Here we can directly see that all cases occur, witnessed e.g. by the Lagrange base polynomial $L(x) := \Pi_{i=0,i\neq j}^{k} \frac{x - x_i}{x_j - x_i}$, which would need a symbol `productincond` in `arith2`.

5 Other Operators that take Qualifiers

There are other operators that take qualifiers in MathML; these are amenable to the same treatment: Consider the naive set $\{x^2 | x < 1\}$, which could be represented in MathML 2 as

```
<set>
  <bvar><ci>x</ci></bvar>
  <condition>
    <apply><lt/><ci>x</ci><cn>1</cn></apply>
  </condition>
  <apply><power/><ci>x</ci><cn>2</cn></apply>
</set>
```

We propose a `suchthatcond` binding constructor that would allow us to write

```
<OMBIND>
  <OMS cd="set4" name="suchthatcond"/>
  <OMBVAR><OMV name="x"/></OMBVAR>
  <OMA>
    <OMS cd="relation1" name="lt"/>
    <OMV name="x"/>
    <OMI>1</OMI>
  </OMA>
  <OMA>
    <OMS cd="arith1" name="power"/>
    <OMV name="x"/>
    <OMI>2</OMI>
  </OMA>
</OMBIND>
```

6 Conclusion

We have studied the representation of extended quantifiers like $\forall x \in S.$ or $\forall x : p(x).$ commonly found in informal mathematical texts. While it is possible to encode these in principle with the quantifiers and connectives provided by the `quant1` and `logic1` content dictionaries from the MathML CD group. The encoding loses the surface structure of the original mathematical expressions and does not support the same intuitive modes of reasoning. Therefore we propose to augment the OpenMath society's set of standard CDs with a new content dictionary `quant3` with six new symbols and `FMP`s that relate them to the classical quantifiers.

By the same concerns for structural adequacy we propose to introduce a symbol for unique existence to the existing `quant2` content dictionary and various operators for "lifted operators" that MathML 2 implicitly supports by allowing them to "take qualifiers".

We feel that the proposed symbols will make the use of OpenMath more intuitive and thus make OpenMath more attractive as a whole for the working mathematician. At the same time, people who prefer the classical quantifiers can refrain from using the new symbols.

It should be noted $\exists!$, $\exists_i$ and $\forall_i$ do *not* require any changes to OpenMath: the rest require an extension of the concept of binding, or the acceptance of mathematically meaningless 'gluing' operators to allow for the fact that bding should take place over both the body and the predicate.

Hence our proposals, in increasing order of scope, are as follows.

1. Adopt `existsuniquelyin`, i.e. $\exists!$.
2. Adopt `existsin`, i.e. $\exists_i$, and`forallin`, i.e. $\forall_i$.
3. Accept that `OMBIND` should be able to bind over more than one child.
4. Adopt `forallcond` etc.
5. Adopt `suchthatcond`, and other related symbols.

Should 3 prove a bridge too far, the subsequent proposals *could* be adopted with a mathematically meaningless gluing operator, as in [DK09].

References

[Bar84] H.P. Barendregt. The Lambda Calculus: Its Syntax and Semantics. *North-Holland*, 1984.

[Con03] World-Wide Web Consortium. Mathematical Markup Language (MathML) Version 2.0 (Second Edition): W3C Recommendation 21 October 2003. `http://www.w3.org/TR/MathML2/`, 2003.

[Con04] The OpenMath Consortium. OpenMath Standard 2.0. `http://www.openmath.org/standard/om20-2004-06-30/omstd20.pdf`, 2004.

[Con08] World-Wide Web Consortium. Mathematical Markup Language (MathML) Version 3.0: W3C Working Draft 17 November 2008. `http://www.w3.org/TR/2008/WD-MathML3-20081117`, 2008.

[DK09] J.H. Davenport and M. Kohlhase. Unifying Math Ontologies: A tale of two standards (extended abstract). In L. Dixon *et al.*, editor, *Proceedings Calculemus/MKM 2009*, pages 263–278, 2009.

Content Dictionaries for Units and Dimensions

James H. Davenport[1] and Jonathan Stratford[2]

[1] Department of Computer Science
University of Bath, Bath BA2 7AY, United Kingdom
J.H.Davenport@bath.ac.uk
[2] IPL Ltd, Eveleigh House, Grove Street, Bath BA1 5LR
Jonathan.Stratford@alumni.bath.ac.uk

Abstract. This paper describes the lessons about OpenMath Content Dictionaries describing units that were learned in the development of the unit conversion tool described in [7, 8], and states explicitly the proposed changes to the OpenMath units-related Content Dictionaries.

1 Introduction

Conversion between different measurement units "ought" to be a simple task, but in practice is surprisingly difficult, leading to catastrophes such as those described in [5, 6], and to hilarious mistakes such as [10]

> Long-term global temperatures are on course to rise by 6C (43F) unless radical changes are adopted in the way that the world produces energy, the International Energy Agency (IAE) said yesterday.

(The true number is 11F: 43F corresponds to 24C. While the error is probably *The Times*'s, the mistake was repeated on the IAE's own website.)

With this goal of defining unit conversions in mind, [4] proposed a set of OpenMath Content Dictionaries (CDs) to describe units, and the associated dimensions. With these CDs come associated Small Type System (STS) files [2], which in the case of units, define the dimension of the units, and thus allow for dimensional consistency.

[7, 8] describe the implementation of a unit converter built on these CDs, and the difficulties, comparatively minor but still worth fixing, that were discovered. The purpose of this paper is to convert this into a concrete list of changes.

Acknowledgements: the authors are grateful to the referees, and to Prof. Kohlhase and Christoph Lange (Jacobs University Bremen), and Bob Dragoset (NIST), for many comments on earlier drafts. M. Tidy (B.Sc. Bath 2009) contributed experience in the area of section 4.

2 Abbreviations and Prefixes

Units have a variety of abbreviations and, particularly in the metric system, a range of prefixes. It is possible, as apparent in [11, section 5.3.5], to regard

prefixed units as units in their own right, and introduce a unit `centimetre` with a formal property relating it to the `metre`, but this way lies, if not actual madness, vast repetition and the scope for error or inconsistency (who would remember to define the `yottapascal`?).

2.1 Prefixes

OpenMath therefore defines prefixes in the `units_siprefix1` CD, with FMPs to define the semantics, e.g. the following one for `peta`.

```
<OMA>
  <OMS name="eq" cd="relation1"/>
  <OMA>
    <OMS name="times" cd="arith1"/>
    <OMI> 1 </OMI>
    <OMA>
      <OMS name="prefix" cd="units_ops1"/>
      <OMS name="peta" cd="units_siprefix1"/>
      <OMV name="unit"/>
    </OMA>
  </OMA>
  <OMA>
    <OMS name="times" cd="arith1"/>
    <OMA>
      <OMS name="power" cd="arith1"/>
      <OMI> 10 </OMI>
      <OMI> 15 </OMI>
    </OMA>
    <OMV name="unit"/>
  </OMA>
</OMA>
</OMOBJ>
```

OpenMath uses a `prefix` operation (described as option 4 of [4, section 4]) to apply prefixes to OpenMath units. Its signature is given as follows.

```
<Signature name="prefix" >
<OMOBJ xmlns="http://www.openmath.org/OpenMath">
  <OMA>
    <OMS name="mapsto" cd="sts"/>
    <OMS cd="units_sts" name="prefix"/>
    <OMV name="dimension"/>
    <OMV name="dimension"/>
  </OMA>
</OMOBJ>
</Signature>
```

which can be seen as

$$\text{prefix} \times \text{unit} \rightarrow \text{unit}. \tag{1}$$

This has the slightly unfortunate property that it would allow, for example, 'millimicrometre', which is explicitly forbidden by [1, p. 122]. This could be solved by making the signature

$$\text{prefix} \times \text{unit} \rightarrow \text{prefixed unit}, \tag{2}$$

which should probably be done.

This construction also allows the use of prefixes with non-SI units, but this is in fact legitimate [1, p. 122].

2.2 Abbreviations

Units also have abbreviations, which are fairly standard in the SI system, less so in other systems. [8] disucssed this at some length, and came to the conclusion that this was *probably* best dealt with outside OpenMath, as the rules are context-sensitive. For example, in `kmph`, the `m` is 'metre', but in `mph` it is mile.

3 Recommendations from [8]

1. Move `litre_pre1964` into a different CD, which is an `official` CD of "obsolete" units. Similar steps should be taken for "obsolete" imperial units.
2. Fix `dimensions1` so as to have a definition for power.
3. Delete `metre_squared` from the `units_metric1`. It is anomalous (why isn't there `metre_cubed`, and why doesn't `units_imperial1` have `foot_squared`?) and tempts a piece of software (such as earlier versions of [7]) into creating units such as

   ```
   <OMA>
     <OMS name="prefix" cd="units_ops1"/>
     <OMS name="deci" cd="units_siprefix1"/>
     <OMs name="metre_squared" cd="units_metric1"/>
   </OMA>
   ```

   which is a deci(metre^2), as opposed to a (decimetre)^2, and is illegal [1, p. 121].
4. `units_imperial1` is missing units such as `inch`, which need to be added.
5. The U.S. units seems somewhat confused — do we need U.S. in the name of the unit as well as in the CD name?
6. U.S. units are missing several items, e.g. gallon (both dry and liquid in theory, though the authors have never seen the U.S. dry gallon in use).
7. All forms of ton(ne) seem to be missing.
8. Add a CD for E.U. units, where different. The only case known to the authors is the `therm`, which comes in both U.S. and E.U. variants. [9, footnote 25] states the following.

Although the therm (EC), which is based on the International Table Btu, is frequently used by engineers in the United States, the therm (U.S.) is the legal unit used by the U.S. natural gas industry.

The difference is about 0.02%.

9. Update all the semantics in the world of OpenMath units so as to adhere to the principles of section 5.1 of [8], in particular definitional[3] numbers should be expressed as elements of **Q**, i.e. as (fractions of) OMI, rather than as floating point numbers.
10. Sort out electrical energy definitions and other suggestions in [4].
11. Modify the signature of `prefix`, as described in section 3.1 of [8], from (1) to (2).
12. Update the definition of `pascal` to include an FMP: currently missing.

Point 9 is more important than it seems: whereas replacing 0.3048 by 3048/10000 might seem pointless, there *is* a difference between 0.5556 (the conversion factor coded in current CDs for Celsius/Fahrenheit conversion) and 5/9, and the conversion factor between `pint_us_dry` and `litre` should be 5506104713575/ /10000000000000, not simply 0.551.

4 A question of time

4.1 The second

In the current CD system, `second`, but no other unit of time, is defined in `units_metric1`. `second`, `minute` etc. are all defined in `units_time1`, along with the associated relationships. There is a Formal Mathematical Property on the definition in `units_metric1`, saying that it is equal to the one in `units_time1`. This should probably be converted to say that it is a formal definition (DefMP), when OpenMath comes to support it. Apart from anything else, this would make it clearer to converter developers that this was *one* unit, not two different but related ones.

4.2 Months and Years

This is a vexed area of definition. The units currently defined in `units_time1` are `calendar_month` and `calendar_year`, with FMPs such as

```
<OMA>
  <OMS name="in" cd="set1"/>
  <OMA>
    <OMS name="divide" cd="arith1"/>
    <OMS name="calendar_month" cd="units_time1"/>
    <OMS name="day" cd="units_time1"/>
```

[3] [8] defines these as "those that started life as experimental, but have since been adopted as architected definitions. An obvious example is '1 yard = 0.9144 metre'".

```
  </OMA>
  <OMA>
    <OMS cd="interval1" name="integer_interval"/>
    <OMI> 28 </OMI>
    <OMI> 31 </OMI>
  </OMA>
</OMA>
```

While correct, and stating that a (calendar) month is always a whole number of days, this does not much help some-one who wants to, say, "convert 5 months into days", though it is certainly arguable that a reasonable response might be "which 5 months?".

We have not looked into this in any detail, but it would be reasonable to add the Julian year (365.25 days), as a precisely defined unit. One could also consider adding, say, the `average_year`, presumably as $365\frac{97}{400}$ days, and the `average_month`, as $\frac{1}{12}$ of this. These might reasonably belong in `units_time1` (or `units_time2`).

Other definitions, such as the sidereal year or the tropical year, are experimental, and indeed time-dependent, quantities, which are areas we have not yet gone into. It would seem reasonable to wait for astronomers to write, or request assistance in writing, such CDs, rather than for us to rush in as amateurs.

It should be noted that these are not the only experimental units. For instance, in [1]'s category of "non-SI units accepted for use with the SI", the electron volt expressed in joules is measured to be $1.602\ 176\ 487(40) \times 10^{-19}$ J. The `(40)` is the standard deviation, in units of the last place, i.e. 40×10^{-28} J. While this *could* be expressed in terms of the `s-dist1` CD, again this is a route down which we have chosen not to rush.

5 Other developments

We should also note that there are other developments in the world of units, which we should take account of. There was a discussion paper on units in MathML [11], but this seems not to have been followed up.

5.1 Collins

There is a paper in MKM 2009, and a companion in OpenMath 2009, discussing a different approach to units and dimensions in OpenMath, and these should be reconciled.

5.2 UnitsML

NIST have launched an OASIS Technical Committee[4] to further the proposed UnitsML (`http://unitsml.nist.gov/`), which we should also track, and preferably be at least compatible with, as two incompatible unit definitions in XML,

[4] `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=unitsml`.

OpenMath/MathML and UnitsML, would be helpful to no-one. However, the first author finds it hard at the moment to see where UnitsML is going, and a lengthy list of units[5] does not seem overly helpful. There is no apparent specification for conversion factors, but it is intended (at least in the U.S.) that they come from the companion database UnitsDB, which is actually outside of the scope of the OASIS UnitsML TC.

OpenMath conversions of the forms described in [4] can be expressed as any OpenMath formula, though in practice they are all $y = \lambda x$, except for the temperature ones, which are $y = \lambda x + \mu$, or the intervals used for time (section 4). λ and μ are OpenMath objects which, following point 9 above, are integers or quotients of integers. Normal UnitsML conversions are of the form $y = d + \frac{b}{c}(x + a)$, which has the advantage that the inverse is $x = (-a) + \frac{c}{b}(y - d)$. More complicated conversions are specified by an external URL[6], or by a SOAP/WSDL call to a remote server[7].

Conversely, UnitsML conversions[8] can be approximate, with an uncertainty radius and the specification of a "last decimal place" (of significance) for each of a, b, c and d, whereas the OpenMath conversions programmed so far are precise, and it is not clear how to express uncertainly of this form in OpenMath (or anything else for that matter: the hard problem is the correlation of errors) [3].

5.3 Others

Many other projects and proposals have "units", as part of them: for example NASA's SWEET[9] set of over 100 OWL ontologies has a "unit" ontology, as well as a "time" ontology. The first author's notes on a recent discussion of unit ontologies in an Ontolog context can be found at `http://staff.bath.ac.uk/masjhd/Sabbatical2009/Ontolog.pdf`.

6 Conclusion

Even without these changes, the OpenMath unit system of CDs was workable[10], and we feel that these changes would improve it further.

Dimensional analysis, supported by the `.sts` files [2], is useful, as it allows a converter to say "you can't convert X to Y", rather than "I don't know how to

[5] `http://unitsml.nist.gov/Schema/Documentation-0.9.17/index.html#Link0529C5B8` seems to be the key reference.

[6] See `http://unitsml.nist.gov/Schema/Documentation-0.9.17/index.html#Link056C85C0`.

[7] See `http://unitsml.nist.gov/Schema/Documentation-0.9.17/index.html#Link05731020`.

[8] `http://unitsml.nist.gov/Schema/Documentation-0.9.17/index.html#Link0526D2F0`.

[9] Semantic Web for Earth and Environmental Terminology (SWEET): `http://sweet.jpl.nasa.gov/`.

[10] Apart from relative temperatures, [7] will operate with them.

convert X to Y". However, it does mean that every unit has to have a dimension, and therefore `volume` is needed to give a dimension to `litre`, for example. It would be possible to split such "derived" dimensions into a separate CD from the SI "base" dimensions, but this was not done in [4], and we have seen no need for it. After all, the SI choice of "base" dimensions is itself arbitrary.

References

1. Organisation Intergouvernementale de la Convention du Mètre. The International System of Units (SI) 8th edition. `http://www.bipm.org/utils/common/pdf/si_brochure_8_en.pdf`.
2. J.H. Davenport. A Small OpenMath Type System. *ACM SIGSAM Bulletin 2*, 34:16–21, 2000.
3. J.H. Davenport and H.-C. Fischer. Manipulation of Expressions. *Improving Floating-Point Programming* (ed. P.J.L.Wallis), Wiley, pages 149–167, 1990.
4. J.H. Davenport and W.A. Naylor. Units and Dimensions in OpenMath. `http://www.openmath.org/documents/Units.pdf`, 2003.
5. Mars Climate Orbiter Mishap Investigation Board. Mars climate orbiter: Phase I report. `ftp://ftp.hq.nasa.gov/pub/pao/reports/1999/MCO\_report.pdf`, November 1999.
6. Wade H. Nelson. The Gimli Glider. *Soaring Magazine*, 1997.
7. J.D. Stratford. OpenMath-based Unit Converter. *B.Sc. Dissertation*, University of Bath, 2008.
8. J.D. Stratford and J.H. Davenport. Unit Knowledge Management. In S. Autexier *et al.*, editor, *Proceedings AISC/Calculemus/MKM 2008*, Springer Lecture Notes in in Artificial Intellgence 5144, Springer-Verlag, pages 382–397, 2008.
9. B. N. Taylor. *Guide for the Use of the International System of Units (SI)*. `http://physics.nist.gov/Pubs/SP811`, may 2007.
10. The Times. Temperature set to rise by 6C, energy agency warns. 13 November 2008, page 62. `http://business.timesonline.co.uk/tol/business/industry_sectors/natural_resources/article5141873.ece`
11. World-Wide Web Consortium (ed. D.W. Harder and S. Devitt). Units in MathML. `http://www.w3.org/Math/Documents/Notes/units.xml`, 2003.

A README from associated CD directory

```
Summary of changes to CDs
dimension1.ocd
     Added relative_temperature
     Added power
units_imperial1.ocd
     Changed conversion factors from OMF to fractions
     (foot, pound_mass, pound_force, degree_Fahrenheit - this was 0.5556)
     Added 'gallon' as 454609/100000 litres
     Made 'pint' exactly 1/8 of gallon
     Deleted miles_per_hour,miles_per_hour_sqrd, as not a new unit
     Added relative_degree_Fahrenheit
```

```
        Added (as exact derivatives of other imperial units)
        inch, furlong, stone, ton_long, ounce, fluid_ounce, acre_foot
    units_imperial_obselete1.ocd
        New CD: rod, pole, perch (three distinct units!); chain; league
    units_metric1.ocd
        Deleted metre_sqrd, metres_per_second, metres_per_second_sqrd
        Deleted Newton_per_sqr_metre
        Defined litre as 1/1000 (rather than 0.001) of m^3
        Deleted litre_pre1964 (see below)
        Added FMP for watt
        Changed 273.15 to 27315/100
        Added relative_degree_Kelvin and relative_degree_Celsuis
    units_metric2.ocd
        New CD: tonne
    units_metric_obselete1.ocd
        New CD: litre_pre1964
    units_siprefix1.ocd
        Reformatted
    units:us1.ocd
        Changed conversion factors from OMF to fractions
        (pint_us_dry; made accurate as 5506104713575/10000000000000, not 0.551)
        pint_us_liquid: now in terms of fluid_ounce_us
        New unit fluid_ounce_us in terms of litres.
        New units cup_us, gallon_us_liquid, gallon_us_dry

    dimension1.sts
        Changed <OMV name="PhysicalDimension"/> to MonoidDimension
        Added relative_temperature with NonMonoidDimension
        Added power
    units_imperial1.sts
        Deleted miles_per_hour,miles_per_hour_sqrd, as not a new unit
        Added relative_degree_Fahrenheit,
        inch, furlong, stone, ton_long, ounce, fluid_ounce, acre_foot
    units_imperial_obselete1.sts
        New CD: rod, pole, perch (three distinct units!); chain; league
    units_metric1.sts
        Deleted metre_sqrd, metres_per_second, metres_per_second_sqrd
        Deleted Newton_per_sqr_metre
        Added relative_degree_Kelvin and relative_degree_Celsuis
        Deleted litre_pre1964 (see below)
    units_metric2.sts
        New CD: tonne
    units_us1.sts
        New unit fluid_ounce_us
        New units cup_us, gallon_us_liquid, gallon_us_dry
```

Integrals and intervals

James H. Davenport

Department of Computer Science
University of Bath, Bath BA2 7AY, United Kingdom
Visiting University of Waterloo
J.H.Davenport@bath.ac.uk

Abstract. The treatment of intervals for integration, summation etc. in OpenMath and MathML leaves something (in fact, quite a lot) to be desired. This paper attempts to analyse the situation, and proposes solutions to OpenMath, and suggestions to MathML.
In particular, we propose a new construct `oriented_interval` to cope with the habit of "integrating backwards". We also propose various other improvements to `interval1`.

1 Introduction

The treatment of "lower/upper limits" in mathematical notation is, if looked at from a purist's point of view, somewhat perverse. There is a split depending on the operator to which they are applied.

$\sum$, $\prod$ Here the interval is almost always an integer interval, or at least discrete. The overwhelming convention is that $\sum_{i=a}^{b}$ assumes $a \leq b$ and i is meant to take the values a, $a+1, \ldots,$ b. The case $a > b$, if mentioned, is (almost) universally taken to be 0 for $\sum$ and 1 for $\prod$, i.e. the unit of the corresponding binary operation.

$\bigcup$, $\bigvee$ **etc.** This is very similar, except that it is rarer to mention the empty case. However, when it is, it is always taken to be the unit of the corresponding binary operation[1].

max**,** min These are very rarely[2] used in the notation $\max_{i=0}^{9}$, rather as $\max_{0\leq i\leq 9}$. The case of a discrete interval is as above, except that there is no clear definition of the empty case. $\pm\infty$ is often used. For a continuous interval (where strictly one should use sup), the case of an empty interval is not defined, and is relatively unnatural to write, i.e. as $\max_{1\leq x\leq 0}$.

$\int$ This is the curious case. What is normally defined in calculus (or at least analysis) texts is $\int_{[a,b]}$, normally under the assumption that $a < b$. One then

[1] There's a conceptual problem for $\bigcap$, since $\cap$ has no unit, since that would have to be the "set of all sets": illegal in ZF. There doesn't seem to be a pragmatic problem though, and OpenMath could solve this if it adopts a proposal circulating from JHD/MK/CAR for "naïve set theory", when the empty $\bigcap$ would be the "universe".
[2] JHD has no examples to hand.

goes on to define the construct $\int_a^b f = \begin{cases} \int_{[a,b]} f & a \le b \\ -\int_{[b,a]} f & a > b \end{cases}$. Hence

$$\int_a^b f = -\int_b^a f. \tag{1}$$

This also lets one write results such as

$$\int_a^c f = \int_a^b f + \int_b^c f \tag{2}$$

without worrying whether $a < b < c$ or not (though one should worry whether the integrals exist at all — see OpenMath recommendation 5).

It is probably worth noting here that an apparent analogue to (2), viz.

$$\sum_{i=a}^{c} x_i = \sum_{i=a}^{b} x_i + \sum_{i=b}^{c} x_i \tag{3}$$

is *not* true, even if $a < b < c$: the correct version is (obviously, with hindsight)

$$\sum_{i=a}^{c} x_i = \sum_{i=a}^{b-1} x_i + \sum_{i=b}^{c} x_i. \tag{4}$$

If one thinks this is purely about continuous/discrete intervals, one should also note that

$$\max_{i=a}^{c} x_i = \max\left(\max_{i=a}^{b} x_i, \max_{i=b}^{c} x_i\right) \tag{5}$$

is only true if $a \le b \le c$.

Why, then, does $\int$ seem so different? One answer, but clearly only a partial one, since the same reasoning would apply to sums etc., is that (2) is appealing and useful. Probably a deeper reason is that one is seeing the beginnings of contour integration here, where (2) is both natural and useful, and the question of whether $a < b < c$ is irrelevant in $\mathbf{C}$.

2 The corpus

This needs more work. However, a Google Scholar search on "where the empty product" yields only that it is defined to be one (or the unit of a more complex structure). We can note the following excerpts from the literature.

1. Landau, in his careful definition of the summation and product symbols, [Lan30, Definition 70] only defines $\sum_{n=y}^{x}$ for $y \le x$.

2. Apostol only defines intervals $[a, b]$ etc. when $a < b$ [Apo67, p. 60], and (2) is his Theorem 1.17, proved when $a < b < c$ at [Apo67, p. 86], and "The proof is similar for any other arrangement of the points a, b, c". However, his Theorem 1.20 states that if $g(x) \leq f(x)$ for every x in $[a, b]$, then

$$\int_a^b g(x)dx \leq \int_a^b f(x)dx,$$

   where $a \leq b$ is implicit. (1) is stated on p. 74.
3. In the constructive domain, [Bis67] defines integration along paths in $\mathbf{C}$, so the question does not arise.

3 MathML and OpenMath currently (version 2)

3.1 MathML

The official W3C recommendation reads as follows.

> The `int` function accepts the `lowlimit`, `uplimit`, `bvar`, `interval`, `condition` and `domainofapplication` schemata. If both `lowlimit` and `uplimit` schemata are present, they denote the limits of a definite integral. The domain of integration may alternatively be specified using `interval`, `condition` or `domainofapplication`. The `bvar` schema signifies the variable of integration. ... For example, the `lowlimit`, `uplimit` pair can be used where explicit upper and lower limits and a bound variable are all known, while an `interval` can be used in the same situation but without an explicit bound variable as in [Con03, section 4.2.3.2]:

```
<apply>
  <int/>
  <interval><cn>0</cn><cn>1</cn></interval>
  <sin/>
</apply>
```

It appears possible for `max` and `min` to take `lowlimit` and `uplimit`, though no examples are given. There is an attribute `closure`, which can be applied (only) to intervals, described as follows.

> [`closure`] indicates closure of the interval. Predefined values: "open", "closed", "open-closed", "closed-open".
> The default value is "closed" [Con03, section 4.3.2.2]

> The `interval` element is used to represent simple mathematical intervals of the real number line. It takes an attribute `closure`, which can take on any of the values "open", "closed", "open-closed", or "closed-open", with a default value of "closed".
> A single `interval` element occuring as the second child of an `apply` element and preceded by one of the pre-defined n-ary operators is interpreted as a shorthand notation for a `domainofapplication`. All other

> uses of an `interval` element as a child of an apply should be interpreted as ordinary function arguments unless otherwise dictated by the function definition. [Con03, section 4.4.2.4.1]

We should note that `lowlimit` has a completely different use in connection with `limit` in MathML.

3.2 OpenMath

Intervals in OpenMath have, surprisingly, no formal properties as such, but merely textual descriptions. That for `interval` is

> A symbol to denote a continuous 1-dimensional interval without any information about the character of the end points (used in definite integration). The arguments are the start and the end points of the interval in that order.

Similarly `integer_interval` is described as follows.

> A symbol to denote a discrete 1 dimensional interval from the first argument to the second (inclusive), where the discretisation occurs at unit intervals. The arguments are the start and the end points of the interval in that order.

OpenMath[3] explicitly adopts (1) in the form shown in section A, but, curiously, only (2) with the guard $a < b \wedge b < c$.

4 Discussion

The MathML 2 usage is arguably consistent, but probably misleading. Since `interval` is explicitly defined to be a subset of the real line, it can be used to replace `lowlimit` and `uplimit` in `int`, but not in `sum`, where one would presumably have to go straight to a `domainofapplication` qualifier. However, `sum` and `product` are said [Con03, section 4.2.3.2] to take the `interval` qualifier, as are the n-ary operators. Presumably as sets, the intervals $(0, 1)$ and $(1, 0)$ are equal, which might surprise some people.

The OpenMath definition is currently almost vacuous. We note that the same form of words is used for `interval` and `integer_interval`, whereas, as we have stated above, usage in ordinary mathematics differs, with integer intervals, as in sum, almost always being taken to be empty if the lower limit is greater than the upper.

[3] `http://www.openmath.org/cd/calculus1.xhtml#defint`.

5 Proposals

5.1 OpenMath

It would be possible to say that OpenMath should move to contour integration. There are two arguments against this, one pragmatic and the other philosophic.

- OpenMath has not yet defined contour integration, and the specification of arbitrary contours looks to be messy.
- Although the inspiration for the use of "oriented" intervals comes from contour integration, $\int_a^b$ is defined purely over the reals, and can be used in places where complex variable theory would have problems with essential singularities, as in $\int_{-1}^{1} \exp(-x^2)\mathrm{d}x$.

Hence we propose the following changes to the CDs in OpenMath. They may as well take effect from OpenMath 2 if they are to tie in with MathML.

1. All the current items "defined" in `interval1` should have proper formal mathematical properties, e.g. for `integer_interval` we would have the following definition, with the corresponding OpenMath given in appendix B,

$$\{n \in \mathbf{Z} | a \leq n \wedge n \leq b\}. \tag{6}$$

   So `interval_oc`, for example, would be

$$\{x \in \mathbf{R} | a < x \wedge x \leq b\}. \tag{7}$$

2. The undifferentiated `interval` should be defined as being any of the specific ones.
3. There should be a new construct in `interval1`[4], called `oriented_interval`. This would be defined to be, in terms of set membership, either `interval_oo`(a,b) or `interval_oo`(a,b), as in appendix C (we use `interval_oo` to avoid problems with interals that start, or end, at infinity).
4. The uses of intervals in `calculus1` be changed to `oriented_interval`.
5. It may be worth considering adding a predicate `integral_defined` to `calculus1`, which would allow one to state various relations with more accuracy.

5.2 MathML

Strictly speaking, it is not up to the current author to make MathML suggestions. However, we note that a variant of the OpenMath suggestions would make MathML's specification less misleading, and also make convergence at MathML 3 easier. We could extend the `closure` attribute with two additional values, `integer` and `oriented`. The former would correspond to OpenMath's `integer_interval`, and would allow `sum` and `product` with `lowlimit`/`uplimit`

[4] Normal OpenMath practice would place this in a new CD. However, JHD thinks that people would not find it there, and it belongs more naturally in `interval1`.

constructs to convert to intervals, which is currently half-suggested. The second would correspond to the proposed `oriented_interval`, and might well even become the default, since that it what seems to be implied by the examples.

There is then a 1–1 correspondence between MathML `interval`s (with `closure` attributes) and OpenMath's `interval1` CD.

References

[Apo67] T.M. Apostol. Calculus, Volume I, 2nd edition. *Blaisdell*, 1967.

[Bis67] E. Bishop. Foundations of Constructive Analysis. *McGraw-Hill*, 1967.

[Con03] World-Wide Web Consortium. Mathematical Markup Language (MathML) Version 2.0 (Second Edition): W3C Recommendation 21 October 2003. `http://www.w3.org/TR/MathML2/`, 2003.

[Lan30] E. Landau. Grundlagen der Analysis. *Leipzig*, 1930.

A OpenMath for (1)

```
<OMOBJ xmlns="http://www.openmath.org/OpenMath" version="2.0"
 cdbase="http://www.openmath.org/cd">
  <OMBIND>
    <OMS cd="quant1" name="forall"/>
    <OMBVAR>
      <OMV name="a"/> <OMV name="b"/>
    </OMBVAR>
    <OMA>
      <OMS cd="relation1" name="eq"/>
      <OMA>
        <OMS cd="calculus1" name="defint"/>
        <OMA>
          <OMS cd="interval1" name="interval"/>
          <OMV name="a"/> <OMV name="b"/>
        </OMA>
        <OMBIND>
          <OMS cd="fns1" name="lambda"/>
          <OMBVAR>
            <OMV name="x"/>
          </OMBVAR>
          <OMA>
            <OMV name="f"/>
            <OMV name="x"/>
          </OMA>
        </OMBIND>
      </OMA>
      <OMA>
        <OMS cd="arith1" name="unary_minus"/>
```

```
        <OMA>
          <OMS cd="calculus1" name="defint"/>
          <OMA>
            <OMS cd="interval1" name="interval"/>
            <OMV name="b"/> <OMV name="a"/>
          </OMA>
          <OMBIND>
            <OMS cd="fns1" name="lambda"/>
            <OMBVAR>
              <OMV name="x"/>
            </OMBVAR>
            <OMA>
              <OMV name="f"/>
              <OMV name="x"/>
            </OMA>
          </OMBIND>
        </OMA>
      </OMA>
    </OMA>
  </OMBIND>
</OMOBJ>
```

B Proposed OpenMath for (6)

```
<OMOBJ xmlns="http://www.openmath.org/OpenMath" version="2.0"
 cdbase="http://www.openmath.org/cd">
<OMA>
  <OMS name="equal" cd="relation1"/>
  <OMA>
    <OMS name="integer_interval" cd="interval1"/>
    <OMV name="a"/> <OMV name="b"/>
  </OMA>
  <OMA>
    <OMS name="suchthat" cd="set1"/>
    <OMS name="Z" cd="setname1"/>
    <OMBIND>
      <OMS name="lambda" cd="fns1"/>
      <OMBVAR> <OMV name="n"/> </OMBVAR>
      <OMA>
      <OMS name="and" cd="logic1"/>
      <OMA>
        <OMS name="le" cd="relation1"/>
        <OMV name="a"/>
        <OMV name="n"/>
      </OMA>
```

```
        <OMA>
          <OMS name="le" cd="relation1"/>
          <OMV name="n"/>
          <OMV name="b"/>
        </OMA>
        </OMA>
      </OMBIND>
    </OMA>
  </OMA>
```

C Proposed OpenMath for `oriented_interval`

```
<OMOBJ xmlns="http://www.openmath.org/OpenMath" version="2.0"
 cdbase="http://www.openmath.org/cd">
<OMA>
  <OMS name="equivalent" cd="logic1"/>
  <OMA>
    <OMS name="in" cd="set1"/>
    <OMV name="x"/>
    <OMA>
      <OMS name="oriented_interval" cd="interval1"/>
      <OMV name="a"/> <OMV name="b"/>
    </OMA>
  </OMA>
  <OMA>
    <OMS name="or" cd="logic1"/>
    <OMA>
      <OMS name="in" cd="set1"/>
      <OMV name="x"/>
      <OMA>
        <OMS name="interval_oo" cd="interval1"/>
        <OMV name="a"/> <OMV name="b"/>
      </OMA>
    </OMA>
    <OMA>
      <OMS name="in" cd="set1"/>
      <OMV name="x"/>
      <OMA>
        <OMS name="interval_oo" cd="interval1"/>
        <OMV name="b"/> <OMV name="a"/>
      </OMA>
    </OMA>
  </OMA>
</OMA>
```

Author Index